

AD-A147 707

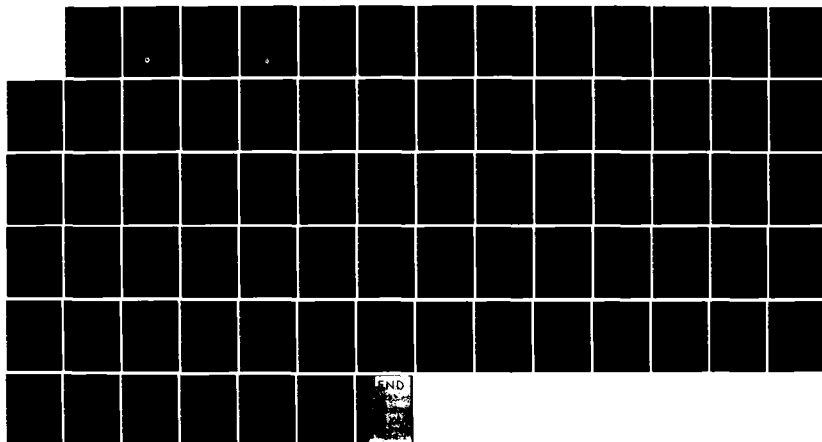
COMPUTER CONTROL OF THE UNDERWATER ACOUSTIC ARRAY HYDRA
(U) DEFENCE RESEARCH ESTABLISHMENT ATLANTIC DARTMOUTH
(NOVA SCOTIA) P R STAAL SEP 84 DREA-TM-84/S

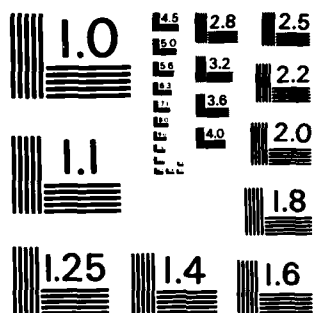
1/1

UNCLASSIFIED

F/G 17/1

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

UNLIMITED

③



National Defence
Research and
Development Branch

Défense Nationale
Bureau de Recherche
et Développement

TECHNICAL MEMORANDUM 84/S
September 1984

AD-A147 707

COMPUTER CONTROL OF THE
UNDERWATER ACOUSTIC ARRAY HYDRA

Philip R. Staal

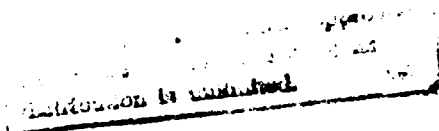
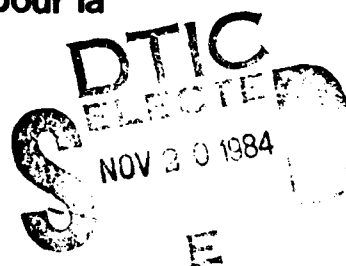
DTIC FILE COPY

Defence
Research
Establishment
Atlantic



Centre de
Recherches pour la
Défense
Atlantique

Canada



84 11 16 027

DEFENCE RESEARCH ESTABLISHMENT ATLANTIC

9 GROVE STREET

P.O. BOX 1012
DARTMOUTH, N.S.
B2Y 3Z7

TELEPHONE
(902) 426-3100

CENTRE DE RECHERCHES POUR LA DÉFENSE ATLANTIQUE

9 GROVE STREET

C.P. 1012
DARTMOUTH, N.É
B2Y 3Z7



National Defence
Research and
Development Branch

Défense Nationale
Bureau de Recherche
et Développement

COMPUTER CONTROL OF THE UNDERWATER ACOUSTIC ARRAY HYDRA

Philip R. Staal

September 1984

Approved by R. F. Brown Director/Underwater Acoustics Division

DISTRIBUTION APPROVED BY

CHIEF D. R. E. A.

TECHNICAL MEMORANDUM 84/S

Defence
Research
Establishment
Atlantic



Centre de
Recherches pour la
Défense
Atlantique

Canada

ABSTRACT

This document describes the software used to control the underwater acoustic array Hydra. The Hydra array was developed at Defence Research Establishment Atlantic (DREA) for bottom mounted use in continental shelf waters. The array is controlled from a minicomputer on board ship, and through a microprocessor in the array. Both the minicomputer software and the microprocessor software are described.

RESUME

Le présent document décrit le logiciel utilisé pour commander le réseau acoustique sous-marin Hydra. Le réseau Hydra a été mis au point au Centre de recherches pour la défense Atlantique (CRDA) pour être installé sur les fonds marins dans les eaux du plateau continental. Le réseau est commandé par un mini-ordinateur à bord d'un bateau et par l'intermédiaire d'un microprocesseur dans le réseau. Le logiciel du mini-ordinateur et le logiciel du microprocesseur sont décrits.

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



Hydra-array control

Table of Contents

Section	Page
Abstract	ii
List of Figures	vi
List of Tables	vii
1. Introduction	1
2. The Hydra Array	1
2.1 Mechanical	1
2.2 Electrical	2
3. General Software Design	4
3.1 Experimental Environment	5
3.2 Hardware Considerations	5
4. Underwater Software	6
4.1 Hardware Configuration	6
4.1.1 Surface Buoy	6
4.1.2 Lower Electronics Unit	7
4.1.3 Hydrophone Interfaces	8
4.2 Microprocessor Software	14
4.2.1 Utility Program UT4	14
4.2.2 Control Program UPCP4	14
5. Shipborne Software	15
5.1 Hardware Configuration	15
5.2 Operating System	15
5.3 Control Program CHYDRA	17
5.3.1 Use of the Control Program CHYDRA	17
5.3.2 Display Format of Array Configuration	18
5.3.3 Internal Format of Array Configuration	18
5.3.4 Interface Gain Control	19
5.4 Data Transfer to Related Software	20
6. Concluding Remarks	23
Appendix	Page
A. Glossary	24

Hydra-array control

B. Instructions for Microprocessor Program UPCP4	26
C. Instructions for Minicomputer Program CHYDRA	27
D. Subroutines Used In Microprocessor Program UPCP4	29
E. Subroutines Used In Minicomputer Program CHYDRA	31
E.1 Input/Output Subroutines	31
E.2 Configuration-editing Subroutines	32
E.3 Numerical-conversion Subroutines	33
F. Microprocessor Array-control Program UPCP4	34
G. Minicomputer Array-control Program CHYDRA	46
References	65

Hydra-array control

List of Figures

Figure	Page
1. A typical deployment of the Hydra array.	2
2. Hydra-array communication and control configuration.	3
3. Hydra lower-electronics-unit configuration.	8
4. Hydra microprocessor memory and registers.	9
5. Hydra hydrophone interface configuration. The square-bracketed letters correspond to commands to the minicomputer program CHYDRA which change the associated parameters.	10
6. Command sequences for controlling the Hydra hydrophone interfaces. . .	13
7. Hydra shipboard hardware configuration.	16
8. CHYDRA log-file and display format. Note that the letters in square brackets refer to commands in CHYDRA used to change values in the corresponding columns.	18
9. File of optimum gain configurations used for Hydra: CHYDRA.GAN	20
10. File format for data transfer to Hydra data acquisition software: CHYDRA.2DA	21

Hydra-array control

List of Tables

Table	Page
1. Hydra control codes	7
2. Hydra data bus command format	11
3. Hydra hydrophone-interface internal addresses	12
4. Data for Hydra hydrophone-interface internal addresses	12
5. Internal format for the set-up of one Hydra hydrophone interface	19
6. Conversion from ASCII character codes to filter and hydrophone status in the Hydra array	22

1 Introduction

This document describes the software used to control the Hydra array. The Hydra array is a modular, digital, hydrophone array developed at Defence Research Establishment Atlantic (DREA). Hydra was developed primarily as a bottom mounted array to collect acoustic information in continental shelf waters. It has been used in combined vertical and horizontal configurations in the waters off Nova Scotia, Newfoundland and England over the past five years.

The Hydra array is a rugged array designed for shallow water use. It normally has sensor spacings from 1.5 m to 63 m, and a total length of less than 400 m. The acoustic frequency response of the array is within the range of 1 Hz to 3000 Hz. The mechanical and electrical aspects of this array have already been described by [Staal, Hughes and Olsen, 1981], but a brief background to the array is presented in Section 2.

The Hydra array system includes not only the array itself and directly connected electronics (the "wet-end"), but also the radio link, the shipboard decommutators, the shipborne control and data acquisition computers, and the high-density backup data recorders. In order to provide the flexibility required to follow swiftly varying experimental requirements, this system is very versatile and complex. It became obvious in the early development of the system that a relatively sophisticated software package would be required to control the array, to display its configuration, and to log automatically the highly variable states of the system.

This software package developed for the Hydra system is described in three sections. Firstly, the rationale for the general software design is discussed in Section 3. Secondly, the software for the microprocessor embedded in the wet-end of the Hydra array system is explained in Section 4. Thirdly, the software for a minicomputer on board the ship is described in Section 5. This minicomputer is usually in control of the whole Hydra array system.

2 The Hydra Array

This section outlines the mechanical and electrical characteristics of the Hydra array system. There has been a need at DREA for a line array, the length and hydrophone spacing of which could be readily altered to suit water-depth and frequency of interest. It was also required that the array be readily adapted to different deployment configurations such as vertical or horizontal, and be reliable, rugged and easy to deploy. The Hydra array was built to meet this need.

2.1 Mechanical

A typical deployment of the array is shown in Figure 1. Some of the array sensors may be held in the vertical by a float and the rest of the sensors may lie on the ocean bottom. The array is anchored by an acoustic release and weight. The sensors in this array are represented by diamond shapes in Figure 1. These sensors are normally hydrophones. Sensor units are connected by interchangeable cable sections, which allows the array length and sensor interspacing to be easily changed. At one end of the array, there is a microprocessor which controls the sensor electronics. This

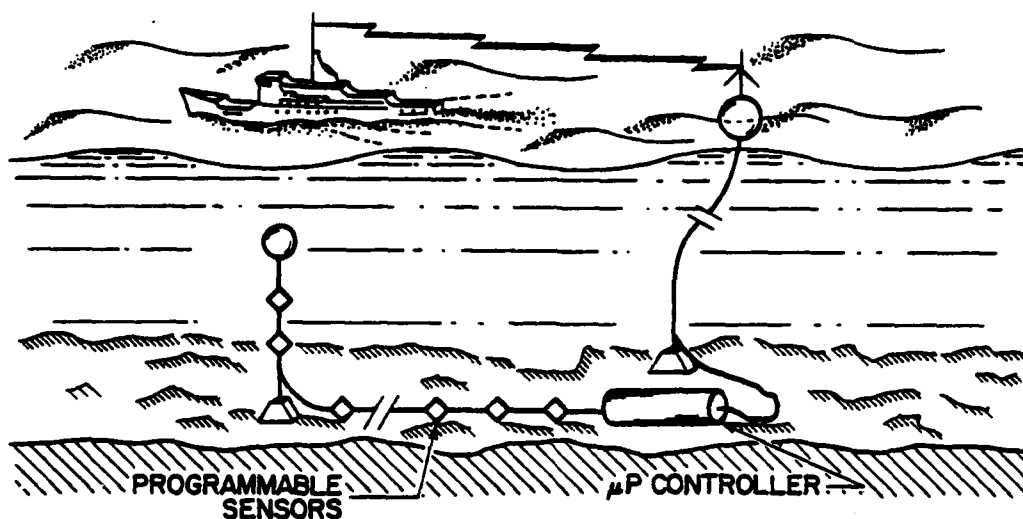


Figure 1. A typical deployment of the Hydra array.

microprocessor unit is linked to a surface buoy by an armoured cable of approximately 700 m length, which carries information between the array and the surface buoy. The surface buoy is tethered to a second weight. Two radio links are used to transmit data to and receive commands from the ship.

2.2 Electrical

The communication and control configuration for the Hydra array is shown in Figure 2. The Hydra array electronics has the form of a digital bus, with hydrophone electronics (or other sensors electronics) connected along it. This array of sensors is controlled at one end by a microprocessor, which in turn is controlled via a radio link by the shipboard minicomputer.

As shown in Figure 2, the entire array system except for the sensor amplifiers and filters is a digital system. The Hydra array system has two modes of operation. The first mode is a control mode, in which array parameters can be changed from the ship. The second mode is a data-acquisition mode, in which data are sent from the sensor electronics to the ship. The mode of operation is normally set by the minicomputer, but the mode can also be set through a simple terminal.

In the control mode, array parameters are loaded by a minicomputer which is on board ship. In this mode, the switches that are shown in Figure 2 are set so that ASCII character data are transmitted from the array to the ship by RS-232 and over the high-speed 168 MHz radio link. At the same time, the minicomputer sends characters over the 27 MHz link from the ship to the wet-end of the array-system. These characters are received in the wet-end of the array-system by the microprocessor and by a simplex

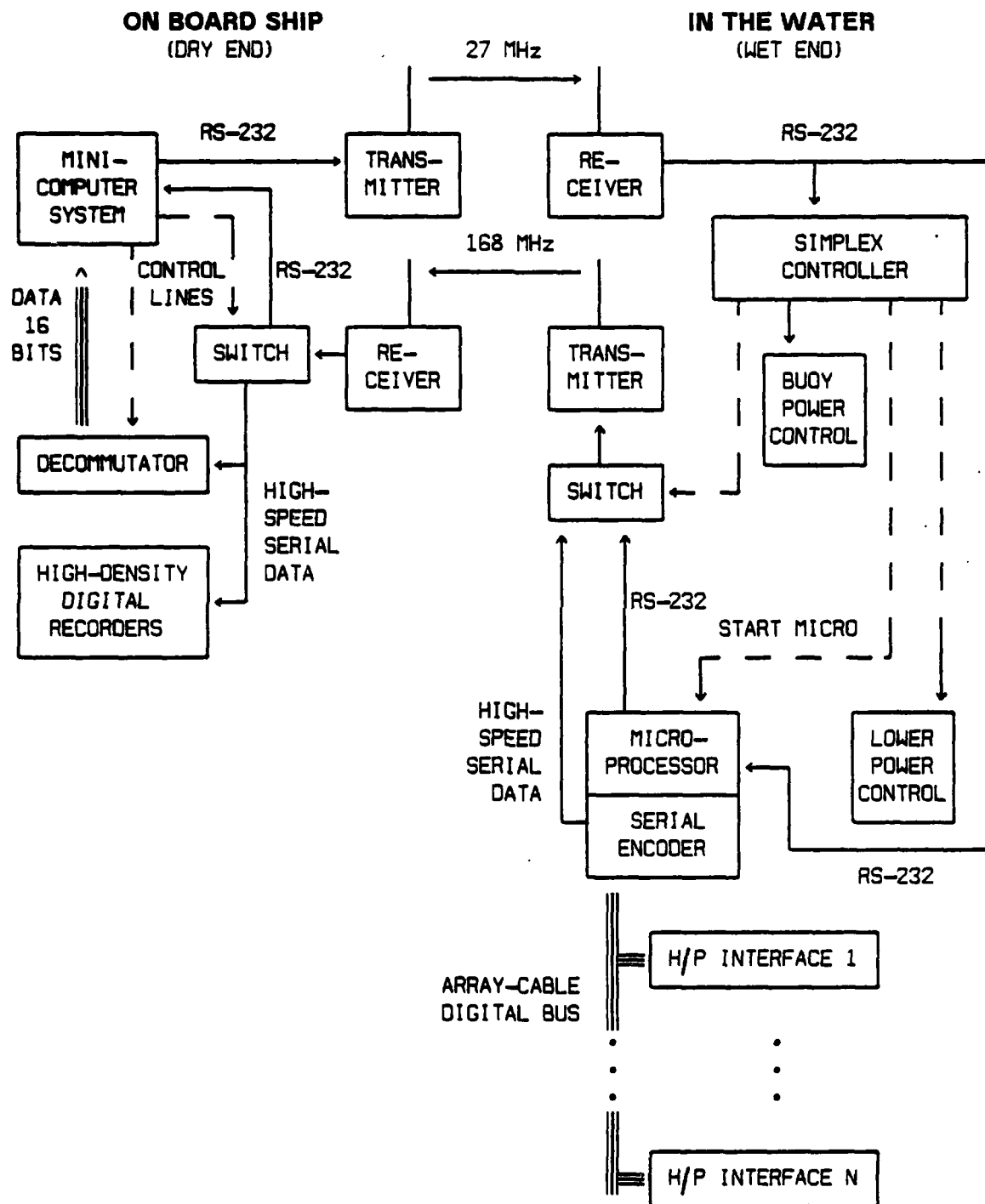


Figure 2. Hydra-array communication and control configuration.

controller. The simplex controller only recognizes a few special control characters. These characters control the power supply for the surface buoy electronics and the power supply for the rest of the array. This simplex controller also controls the mode of operation of the wet-end of the array system and it is used to restart the microprocessor. Characters other than the special control characters are interpreted by the microprocessor. The characters which are sent from the minicomputer to the microprocessor can be used to load a program into the microprocessor from the minicomputer. They can also be used to send information which can be passed on by the microprocessor to the hydrophone interfaces. The microprocessor, as well as receiving characters, can send characters over the 168 MHz radio link back to the minicomputer. The characters received by the minicomputer are used to check that the microprocessor is operating properly and that it is receiving commands correctly.

After turning on the power in the wet-end of the array system, the whole system is left in the control mode, and a program is loaded into the microprocessor. This program is then run, and information is passed from the minicomputer, over the radio link, through the program in the microprocessor, to the appropriately addressed hydrophone interfaces. The designated hydrophones have their associated gains, frequency responses and other characteristics adjusted. When these hydrophone interfaces are properly set, the mode of the Hydra array system is switched to data acquisition. This switching is initiated at the minicomputer which then generates commands to the simplex controller. The simplex controller then calls upon the microprocessor electronics to initiate data acquisition in the hydrophone interfaces.

In the acquisition mode of operation, acoustic data are transmitted at a constant high-speed rate from the array to the ship. The array parameters cannot be changed without interrupting this mode with a command to the simplex controller. Acoustic signals received at the hydrophones are amplified and digitized in the hydrophone interfaces. This design feature avoids crosstalk and phasing problems found in twisted-pair analog systems. When the hydrophone interfaces are switched into data-acquisition mode, those which are not set up for use turn their own power off. The operational interfaces start sending their data up the array-cable bus to the serial encoder. A parallel bit format is used. These data are sent up the bus at times determined by a sequence programmed into the hydrophone interfaces. The serial encoder converts the parallel data words to a high speed serial form. Each hydrophone output is multiplexed into the serial sequence together with injected synchronization words. This serial sequence is then sent up the cable to the surface buoy, and transmitted via the 168 MHz data link to the receiver on board ship. From the ship-board receiver, the data pass through a switch to the high-density digital recorders and also to a decommutator which provides 16-bit parallel data to the minicomputer for on-line analysis.

3 General Software Design

The control software which is described in this document was written to suit the Hydra array hardware outlined in the previous section, and to suit the environment in which this hardware is used. Thus, a brief description of the influence of the experimental environment and of the hardware on the software will help in understanding the software which is described in the rest of this document.

3.1 Experimental Environment

The way that experimental trials are conducted and the way that an acoustic array is deployed in shallow water strongly influenced both the hardware and software design of the Hydra array. The Hydra array is typically deployed with hydrophones on the bottom and in the water column. The hydrophones can be in vastly different acoustic fields. Both the frequency content and the overall signal amplitude can vary strongly from sensor to sensor, making it necessary to have independent control over filters and amplifiers for each hydrophone. Not only can hydrophone signals be vastly different from hydrophone to hydrophone, but the received signals can also change very quickly with time. A typical experiment may consist of dropping explosive charges every few minutes, alternating between shallow and deep charges. The array response characteristics may need to be changed to match each charge. Another consideration is that sensors other than hydrophones may be required, such as tilt/pitch meters to determine the straightness of a vertical array or seismometers to receive shear waves. The software must be designed to make control of such additional sensors easy. Furthermore, the physical deployment of the array is often changed during a trial. Such changes necessitate flexibility in the controlling software. Changes are usually due to different experimental requirements, but occasionally may be due to equipment failure.

The flexibility just described has a cost in the need to keep track of the complex states of the Hydra system. In normal use, a different set of hydrophones with different gains and frequency characteristics may be turned on every few minutes. This reconfiguration is done at the same time that the scientists, programmers and technicians are very busy with other duties. From experience in the early development of the Hydra system, it was found that hand kept logs of the system configuration were often spotty and inaccurate. All the data from sea trials are recorded on high-density digital tape recorders. These recordings contain no system information. Therefore, to keep accurate records of the state of the Hydra system, automatic logs are kept by the controlling minicomputer. Also, the state of the system is automatically recorded along with any acoustic data recorded by the minicomputer. A record of all previous system changes can be printed at anytime.

3.2 Hardware Considerations

The hardware used in the Hydra array sets constraints on the software implementation. Since the wet-end of the array system is battery powered, we chose a microprocessor with low power consumption. At the time of construction, the only microprocessors available with low power consumption had limited speed and capabilities. There was no cross-assembler available for the one we chose until we wrote our own. Thus, the microprocessor's functions are somewhat limited. For efficiency in programming, a high level language (Fortran) is used on the minicomputer to do most of the control, but for reliability, some control is available using software in the microprocessor only. This means that the minicomputer system in Figure 2 can be replaced by a simple terminal for controlling the array. Thus, data collection can continue normally, even if the minicomputer system fails. As well, a technician can control the array for test purposes without tying-up a minicomputer. Because of the hardware design, the software in the microprocessor can be downloaded from the minicomputer. This means that the minicomputer can completely redefine the operation of the wet-end of the array system.

The use of a radio link to pass information between the array and the ship also sets

constraints on the software. For reliability, each character sent to the wet-end of the array system from the minicomputer is interpreted by the microprocessor in the wet-end and then repeated to the minicomputer. The minicomputer compares what it sends with what it receives back, in order to make sure that its messages are received correctly. In the duplex mode which is used for control the data rate is 300 baud, which is quite slow. However, the baud rate of the link is sufficiently fast that when the operator has configured the array operating parameters to his liking and uses the *transmit* command, the entire setup of *every* parameter in the array can be sent from the minicomputer to the microprocessor and implemented in a very reasonable period of time. Approximately 3 seconds, plus 0.3 seconds per hydrophone in use is required. Approximately 95 per cent of this time per hydrophone is for radio data transmission. The complete setup of an array of 16 hydrophones takes approximately 8 seconds, during which time the array is not sending acoustic data to the ship. When a simple terminal (rather than the minicomputer) is used for system configuration, the process takes rather longer - roughly 4 minutes, assuming no errors are made. For a typical experiment, with only a few minutes between gain and filter changes, these changes might not be made fast enough using only the simple terminal. An option that has been used to beat the 300 baud speed limit, is to load a special program into the microprocessor. This program keeps track of the time, and at predetermined times makes changes to hydrophone interface parameters such as gains and filters. This type of program has been used for test purposes: cycling through all states of the array at a high speed. Another limit of the present link is the 655 kbit/second data rate for digitized data being sent from the wet-end of the array system to the shipboard electronics. This limit requires tailoring of the array configuration to match both the experimental requirements and the available data-transmission bandwidth. The Hydra system is flexible enough to allow this tailoring, since one can control the number of sensors turned on, the digitizing rate, and the number of bits per word.

4 Underwater Software

The microprocessor in the wet-end of the Hydra array system requires its own software. A description of the hardware connected to this microprocessor is included in the following section since this hardware strongly influences the software design.

4.1 Hardware Configuration

The hardware in the wet-end of the Hydra array system can be conveniently discussed in three parts. Part one consists of the surface buoy, which communicates with the ship-board electronics over the radio link and communicates down a cable to the lower electronics unit. Part two is the lower electronics unit, which contains the microprocessor and serial encoder. Besides communicating with the surface buoy, the lower electronics unit communicates with the hydrophone interfaces along the array-cable digital bus. Part three consists of the several hydrophone interfaces, which are all essentially identical.

4.1.1 Surface Buoy

The surface buoy contains radio communications equipment, a simplex controller,

and a power supply. The simplex system is used to control a few critical functions of the wet-end of the Hydra system. These functions are listed along with their codes in Table 1. The simplex controller checks for these codes in the normal serial ASCII data stream from the ship. As shown in Table 1, the simplex controller can be used to turn on and off all the power in the wet-end of the Hydra system except for the simplex system's own power. The power can therefore be remotely switched off to save battery power when the array is not being used. The mode of the radio link can also be set. If *transmit ASCII* is selected, then the radio link is switched to enact full-duplex ASCII communication between the ship system and the microprocessor. If *transmit data* is selected, then the radio link is switched to enable high-speed data transmission from the array to the ship. Finally, the simplex system can reset and run the utility program UT4 (described in Section 4.2.1) in order to "boot-strap" the microprocessor. This is done by using the utility program UT4 to bring software into the microprocessor's memory and to start this software.

Table 1

Hydra control codes

<CONTROL> - J	POWER ON SURFACE-BUOY ELECTRONICS
<CONTROL> - X	POWER OFF SURFACE-BUOY ELECTRONICS
<CONTROL> - I	POWER ON SUB-SURFACE ELECTRONICS
<CONTROL> - \	POWER OFF SUB-SURFACE ELECTRONICS
<CONTROL> - T	TRANSMIT DATA FROM ARRAY TO SHIP
<CONTROL> - V	TRANSMIT ASCII FROM ARRAY TO SHIP
<CONTROL> - R	RESET MICROPROCESSOR, RUN UTILITY PROGRAM

4.1.2 Lower Electronics Unit

The lower electronics unit contains the microprocessor and a serial encoder. As shown in Figure 3, the microprocessor is in communication with the ship by an RS-232 serial duplex link, and in communication with the array over a parallel data and command bus. The RS-232 link is used to control the microprocessor; however, this link is in turn controlled by the simplex system.

The microprocessor has control of the hydrophone interfaces over the parallel data bus, and it also has control of the serial encoder. The serial encoder takes parallel data words from the hydrophone interfaces and converts them to a serial stream of data bits including synchronization bits. The serial encoder also produces sampling and calibration signals to be sent to the interfaces. As shown in Figure 3, there are seven control ports which the microprocessor uses to set parameters in the serial encoder. The number of data channels being sent to the ship can be selected by ports 1 and 5. In normal operation, the same number must be output to both ports. A calibration signal can be sent down the bus to the hydrophone interfaces by port 2. The digitization rate to be used at each hydrophone can be selected by port 3. Digitization rates of 8192, 8192/2, 8192/3, 8192/4 etc. (Hz) can be selected. Port 4 is used to directly control the hydrophone interfaces. The number of bits per word can be set in the range from 6 to 12

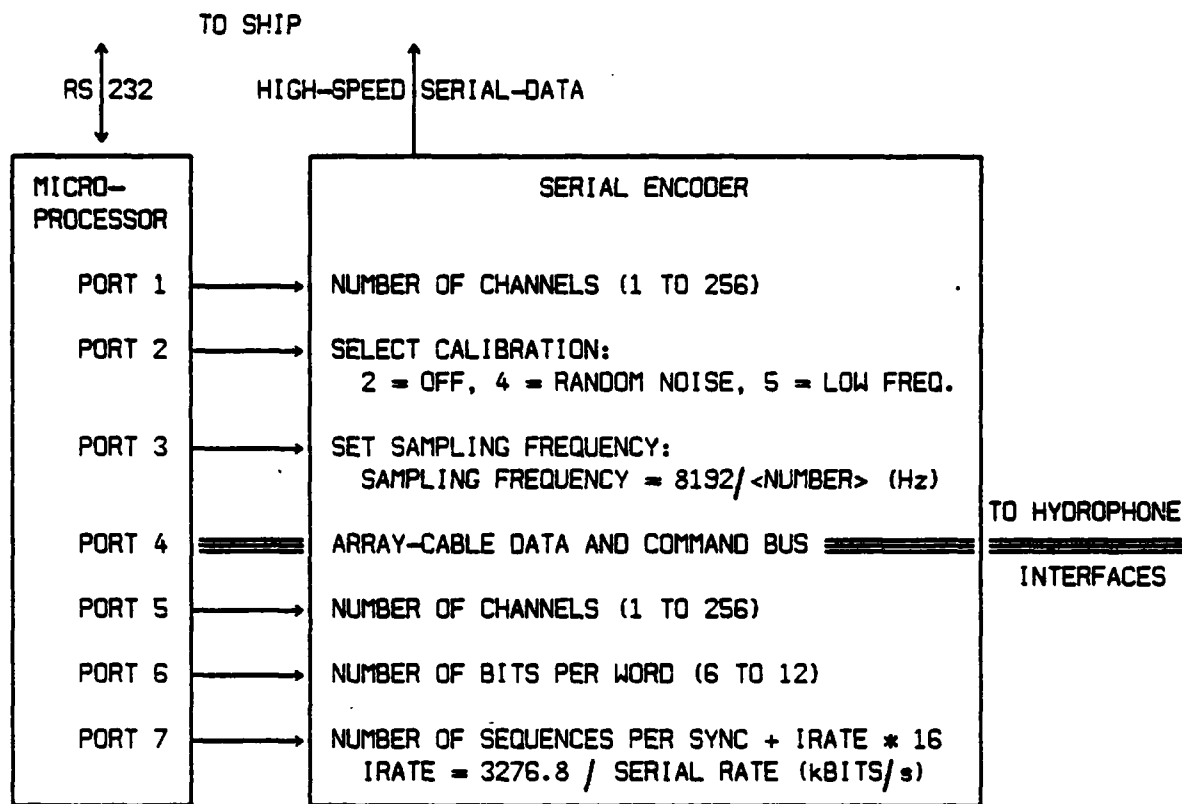


Figure 3. Hydra lower-electronics-unit configuration.

by port 6. Port 7 sets the number of data sequences of all channels before a synchronizing word is sent along the serial data stream.

The memory that the microprocessor can access is shown in Figure 4. There are sixteen 16 bit registers split into bytes. There are 4k bytes of read/write memory (RAM) currently built into the system. The utility read-only memory (ROM) is higher in the address space, with a small amount of read/write memory just above the ROM for use by the utility program.

4.1.3 Hydrophone Interfaces

The microprocessor controls interfaces which are connected to the array-cable data and command bus. These interfaces control many characteristics of the way that the hydrophone signals are conditioned and digitized as shown in Figure 5. The logical number of an interface can be set. This number determines when the interface sends its data back to the ship. In normal use, acoustic signals are received by the hydrophone, passed through a high-pass filter, amplified, passed through a low-pass anti-aliasing filter, digitized, and sent on the array-cable data bus to the serial encoder. The high-pass filter characteristics can be changed to accommodate the low-frequency acoustic

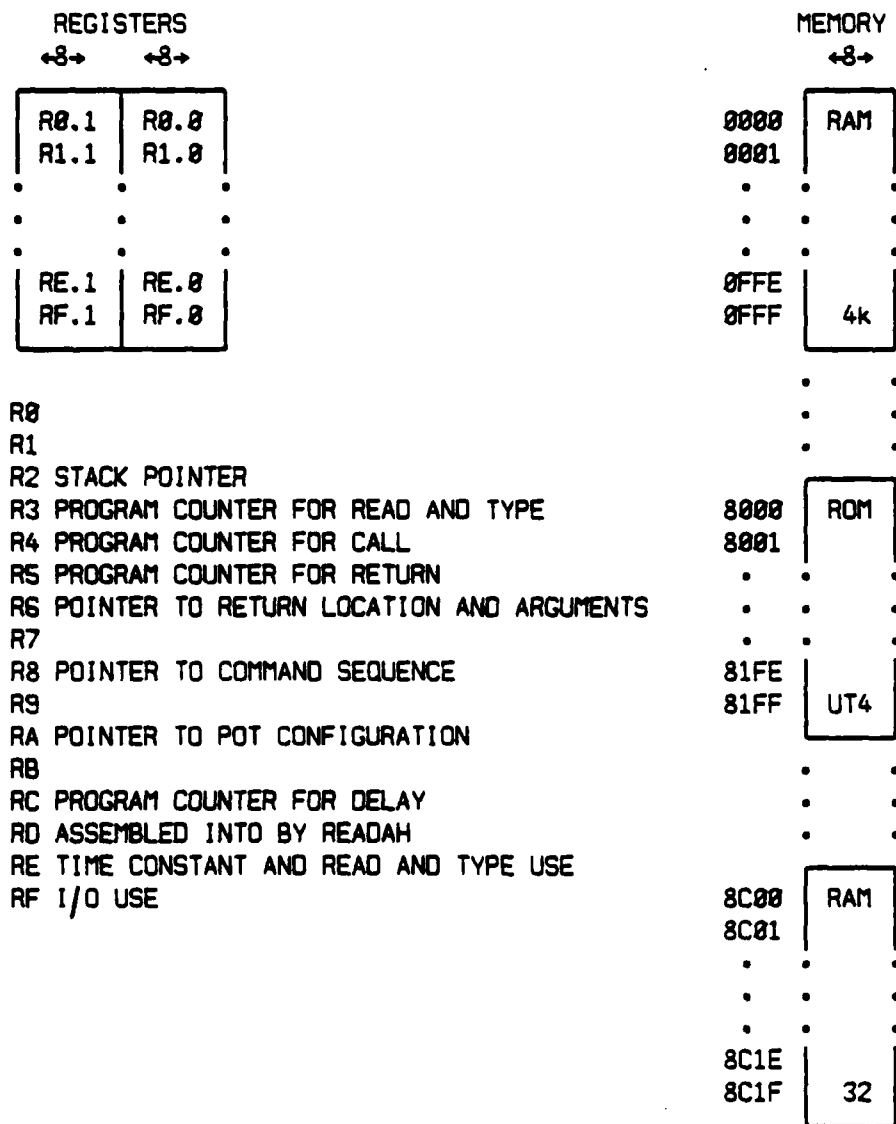


Figure 4. Hydra microprocessor memory and registers.

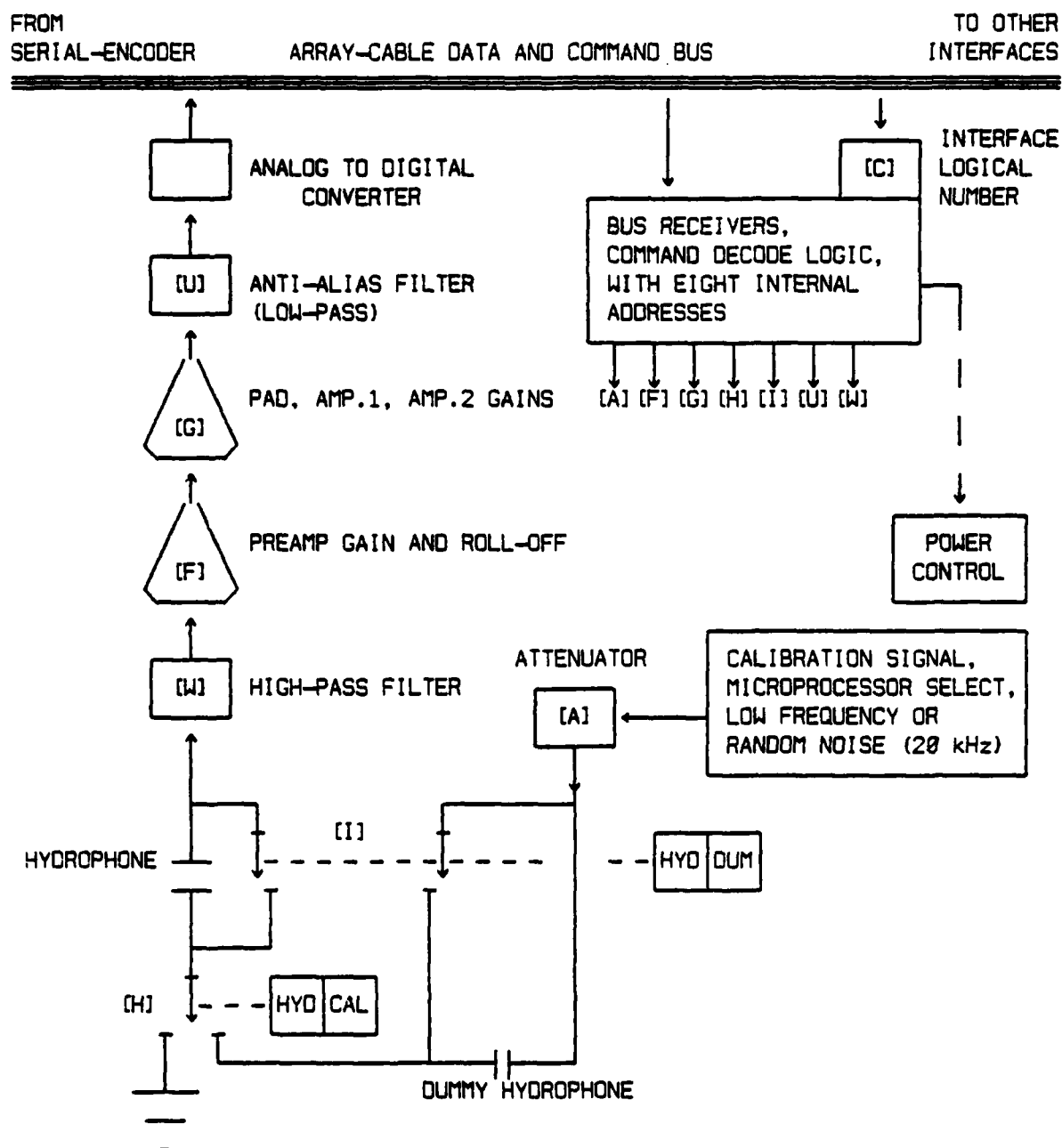


Figure 5. Hydra hydrophone interface configuration. The square-bracketed letters correspond to commands to the minicomputer program CHYDRA which change the associated parameters.

noise characteristics. The low-frequency noise levels vary strongly depending on the sea-state, the hydrophone locations and local shipping. The amplifier gains can be changed to match the overall signal and noise levels. The anti-aliasing filter can be switched out in order to digitize high frequency signals such as acoustic release commands or pings from the ships echo sounder. The acoustic pings from the acoustic release can be used to determine the positions of the hydrophones. In addition to normal data collection from the hydrophone, calibration signals can be internally generated, selected, attenuated and injected past or through the hydrophone to check each individual hydrophone and/or interface.

The format of commands which are sent along the array-cable data and command bus to the interfaces is shown in Table 2. The two high-order bits (7 and 6) define four types of command. The bottom three commands in Table 2 are described first. The address in these commands is the address hard-wired into the individual interface electronics board. The interfaces are also numbered with logical numbers by the *set address* command to determine in what order they will output data back to the ship. The logical number for the interface addressed with this command follows as the next byte on the bus. In order to enter other parameters into an interface, it is opened for writing with the *set Q* command. When all required parameters are entered into the interface, the interface is closed to writing with the *reset Q* command. The last form of command, at the top of Table 2, writes data into a specific location in an interface to control gains, etc.. This command has zeros for the two highest order bits, a three bit internal-interface address, and a three bit data word to be entered into that internal address. The internal addresses are listed in Table 3 and the consequences of entering data into those addresses are listed in Table 4. Typical command sequences used by the microprocessor for controlling the hydrophone interfaces using the command format just described are shown schematically in Figure 6. The two command sequences shown in this figure are used by the manually operated commands of the program UCP4 (described in Section 4.2.2). There are delays after each command to an interface to allow it to accept the command. The hexadecimal command FF sent to the bus clears it. All the rest of the commands are those listed in Table 2.

Table 2

Hydra data bus command format

7	6	5	4	3	2	1	0	
0	0	X	X	X	X	X	X	DATA TO BE ENTERED INTO H/P INTERFACE
								DATA ADDRESS
0	1	X	X	X	X	X	X	SET ADDRESS: NEXT BYTE IS LOGICAL H/P NUMBER
1	0	X	X	X	X	X	X	SET Q: ADDRESS THIS INTERFACE FOR DATA ENTRY
1	1	X	X	X	X	X	X	RESET Q: STOP ACCEPTING DATA INTO THIS INTERFACE
								HARD-WIRED ADDRESS OF INTERFACE

Table 3

Hydra hydrophone-interface internal addresses

2 1 0	
0 0 0	AMPLIFIER 1
0 0 1	AMPLIFIER 2
0 1 0	ATTENUATOR
0 1 1	CALIBRATION-SWITCH/PAD WITH R-C FILTER
1 0 0	PREAMPLIFIER
1 0 1	DUMMY-H/P, LOW-PASS-FILTER
1 1 0	CALIBRATION-SWITCH/PAD WITHOUT R-C FILTER
1 1 1	UNUSED

Table 4

Data for Hydra hydrophone-interface internal addresses

DATA 5 4 3	AMP1,2 (dB)	PREAMP (dB)	ATTEN. (dB)	CAL/PAD (dB)	HYD. OR DUMMY, LOW-PASS IN OR OUT
0 0 0	42	42	48	18, HYD	HYD, L.P. IN
0 0 1	36	36	42	36, HYD	DUM, L.P. IN
0 1 0	30	30	36	0, HYD	HYD, L.P. OUT
0 1 1	24	24	30	-	DUM, L.P. OUT
1 0 0	18	18	24	18, CAL	HYD, L.P. IN
1 0 1	12	-	18	36, CAL	DUM, L.P. IN
1 1 0	6	-	12	0, CAL	HYD, L.P. OUT
1 1 1	0	-	6	-	DUM, L.P. OUT

The current versions of hardware and software allow a maximum of 64 sensor interfaces. The array-cable data and command bus is a 12 bit bus, but only 8 bits are presently used for commands, as shown in Table 2. Two of these 8 bits are used for defining the type of command, leaving 6 bits for the hard-wired addresses of the interfaces.

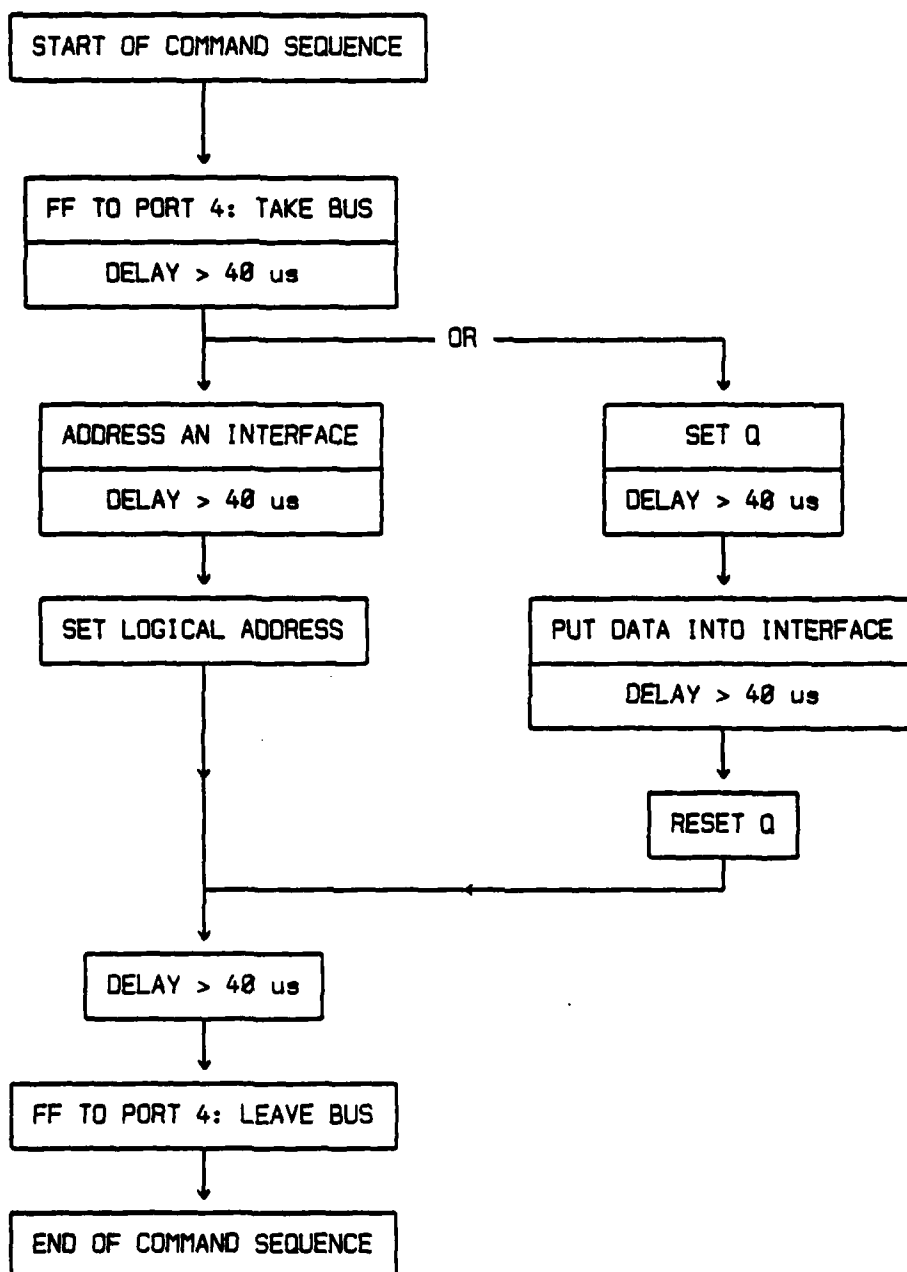


Figure 6. Command sequences for controlling the Hydra hydrophone interfaces.

4.2 Microprocessor Software

The software in the wet-end of the Hydra array system consists of a commercially available general-utility program and a program designed specifically for controlling the Hydra array.

4.2.1 Utility Program UT4

A small package of software called the *utility program* or UT4 is used in most microprocessor operations described here. This utility program is in read-only memory as shown in Figure 4, which makes the program available as a boot-strap to the microprocessor, and also makes it invulnerable to data-link problems. The utility program is used for all serial input/output operations of the microprocessor, for downloading software into microprocessor read/write memory, and for some microprocessor program timing. The commands available with UT4 through the microprocessor's RS-232 input are listed below:

?M<address> <count><CR>	Lists the contents of <count> bytes of memory starting at <address>.
!M <address> <data><CR>	Puts <data> into memory starting at <address>, where <data> consists of a hex pair for each byte.
\$P<address><CR>	Starts program execution at memory location <address>.

As well as running UT4 directly, some of its subroutines are called from the control program UCP4. The subroutines in UT4 that are of use when programming the microprocessor are listed below:

CALLING SEQUENCE	NAME	FUNCTION PERFORMED
SEP \$C .BYTE <delay number>	DELAY	Delay for a time given by <delay number>.
SEP 4 .WORD \$813B	READAH	Read an ASCII character into register RF.1. If a hex digit, then hex value put in register D and DF = 1. If not a hex digit, then DF = 0.
SEP 4 .WORD \$81A4	TYPE	Type the ASCII character from register RF.1.
SEP 4 .WORD \$81A2 .BYTE <character>	TYPE6	Type the ASCII character <character>.

Note that all reading and typing in the above subroutines is done through the microprocessor's RS-232 input/output connection.

4.2.2 Control Program UCP4

The microprocessor program which is currently used for control of the Hydra array is called UCP4. It has code which allows the array to be controlled with a simple terminal.

The program also has code which allows a minicomputer on the ship to do the controlling. This program is downloaded from the ship, usually from the minicomputer, but sometimes from a terminal with a tape cassette reader. Tape cassettes have been prepared with pre-recorded programs.

The commands for the control program UCP4 are listed in Appendix B. The minicomputer uses the utility program UT4 to load an array configuration table from the minicomputer into the microprocessor's memory, and to start the program UCP4. The minicomputer then uses command Z in UCP4 to set up the entire array from that table. The minicomputer also uses the command UT4<CR> to return to the utility program. The remaining commands in this control program are for use from a terminal on the ship.

The subroutine calls which are used in the control program UCP4 are listed separately in Appendix D. These calls handle terminal input/output, menu-branching and output to the array-cable data and command bus. They are also listed along with the main body of the program in Appendix F.

5 Shipborne Software

The main control software for the Hydra array resides in a minicomputer on board the ship. Since the control software is influenced by the hardware and operating system environment, this environment is described first. The control software obtains information about the array configuration from the operator while being used to control the array. This information is then passed on to other programs in the minicomputer. The information transfer is described at the end of this section.

5.1 Hardware Configuration

The shipboard control hardware, shown in Figure 7, consists of a minicomputer which is in communication with the wet-end of the array system using a radio transmitter and receiver, a terminal which displays the current set-up of the array, and a console terminal to control the minicomputer. The minicomputer has peripherals including a time code reader/generator, a removable 5 Mbyte disk, an 80 Mbyte fixed disk, and two 125 inch/s 9-track tape-drives. The time code reader/generator is used as the master time reference for all data gathering. The data recording hardware is also shown in Figure 7. The decommutator-box converts the high-speed serial data from the array into parallel 16-bit words which are sent to the minicomputer for analysis. The decommutator-box also provides analog reconstruction of the digital data from the array. The decommutator-box is internally controlled by a microprocessor running a BASIC interpreter. This microprocessor is controlled by the minicomputer, but its operation will not be described in this document. There are also high-density-digital-tape recorders which are capable of continuous recording at 655 kbits/s for fourteen hours without attention.

5.2 Operating System

The minicomputer software runs under the Digital Equipment Corporation's RT-11 real-time operating system. This operating system is used for almost all input/output from the program CHYDRA (described in Section 5.3). One of the disks connected to the

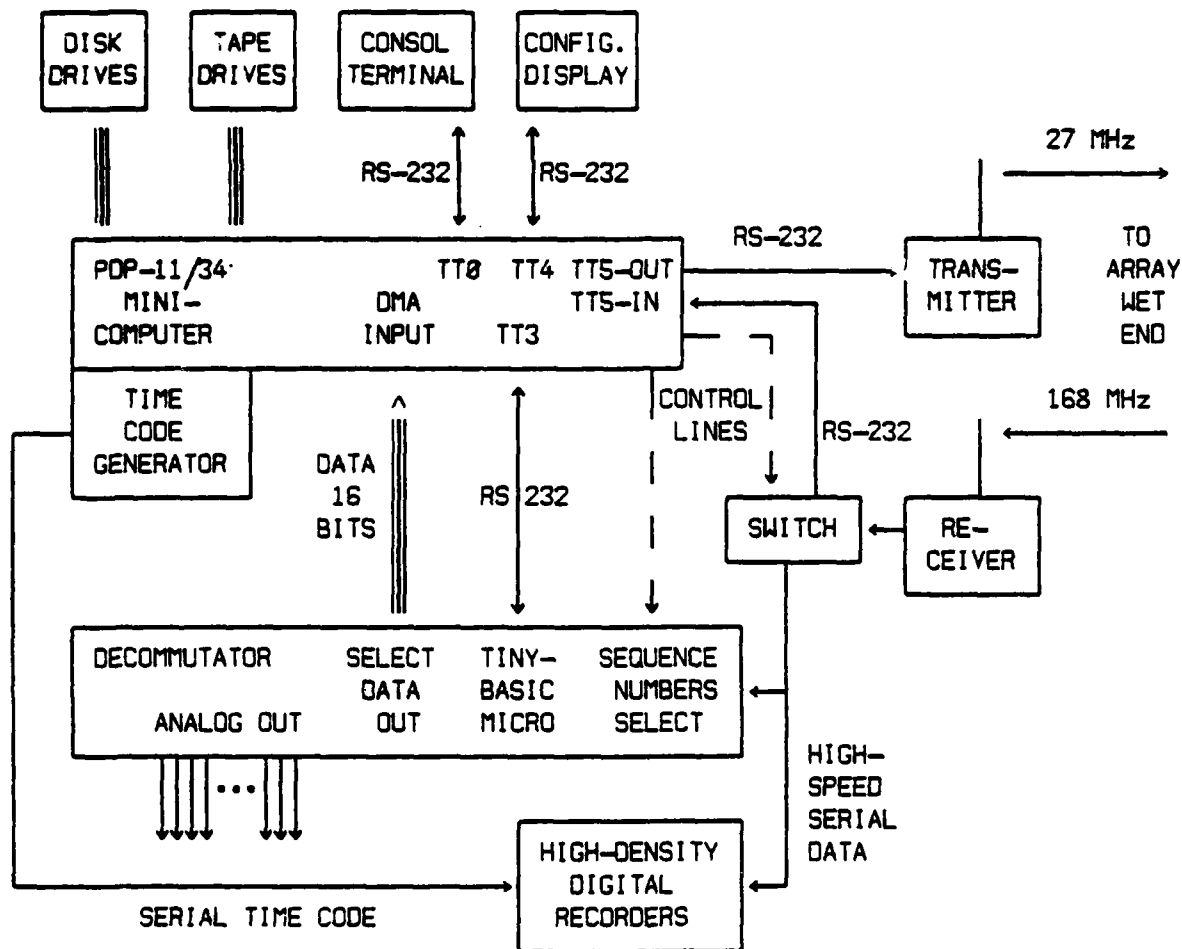


Figure 7. Hydra shipboard hardware configuration.

minicomputer is designated as the "operating-system disk". The files used by program CHYDRA are kept on the minicomputer's operating-system disk, and software to be transferred to the microprocessor by CHYDRA may be kept on any of the minicomputer's storage devices.

There are a few operating system commands that may be required while operating the Hydra array. In the following description, what is typed by the user is bold, and what is typed by the computer is in *italics*. <CR> means carriage-return. When the minicomputer is first turned on or restarted by pressing the front-panel "boot" button, a diagnostic code and a \$ prompt appear on the console. Respond to the prompt as shown below to get RT-11 started:

\$DB<CR>

The system will start, give some initial messages, then wait for commands while displaying a . prompt. Some of the possible commands are:

.R CHYDRA<CR>	Runs the control program CHYDRA.
.PRINT SY:CHYDRA.LOG<CR>	Prints past system configurations.
.RENAME SY:CHYDRA.LOG <new filename>	To save an old log file.

See the PDP-11/34 and RT-11 user manuals for further information.

5.3 Control Program CHYDRA

The program which is normally used for overall control of the Hydra array is called CHYDRA, and is listed in Appendix G. This program controls the Hydra array, displays its configuration on a video terminal, passes the array configuration to succeeding programs, records a log of configurations used during a trial, and allows the user to print the current configuration on a line printer.

The CHYDRA program consists of a mainline routine which provides a menu of functions for the operator, and 22 subroutines to carry out those functions. The subroutines can be divided into several classes: input/output subroutines, configuration-editing subroutines, and numerical-conversion subroutines. These subroutines are described in Appendix E.

5.3.1 Use of the Control Program CHYDRA

The control program CHYDRA is used regularly during a typical trial. CHYDRA is used to load the software into the wet-end of the array system before deployment, and to set up the array for monitoring during deployment of the array. For monitoring ambient noise, CHYDRA is used to tailor the response of the array to the ambient noise received at each sensor. The relatively-large acoustic dynamic range of the array and the slow temporal change in ambient noise usually only necessitate changes in array response on a scale of hours. For monitoring sound arrivals from explosive sound sources, gain changes may be required as frequently as every few minutes.

At some time in a trial, the array may be set up for calibration. Precision levels of pseudo-random electrical noise are injected in series with each hydrophone, controlled through the CHYDRA program. The gain settings to be calibrated are selected and the resulting calibration signals are recorded like normal acoustic data. These calibration signals may be used as diagnostic tools to reveal faulty components of the array.

The hydrophone positions may also be localized with the help of the CHYDRA program. The anti-aliasing filters may be bypassed using CHYDRA commands so that pings from the acoustic releases can be recorded. These pings can give ranges from the acoustic releases to the hydrophones by several paths. Knowing these ranges along with ranges from another source (usually long-range explosive sounds) allows triangulation to determine the hydrophone positions.

After completion of a trial, the log file or files that were created by CHYDRA are printed for subsequent reference.

The actual command sequences used to do the typical operations described in the four previous paragraphs are described in Appendix C.

5.3.2 Display Format of Array Configuration

One display format of the array configuration is used for the console-terminal, the configuration display, printer output and for the log-file. This format is shown in Figure 8. The date and time at which the array is set-up are recorded from the master time-code generator, along with the version date of the CHYDRA program. The command codes necessary to change the listed values are given in the display [enclosed in square brackets], for quick reference. All the CHYDRA commands ask for a beginning pot number and an ending pot number so that large numbers of hydrophone interfaces can be changed at the same time in order to speed up editing of the configuration table.

```
DATE: 83- 6-29,      TIME: 12:34:31.463,      CHYDRA version 83-5-16
```

POT #	WIRED #	PRE-AMP GAIN(dB)	PAD, G1, G2 GAIN(dB)	HIPASS FILTER	AALIAS FILTER	HYD CAL	HYD DUM	ATTEN (dB)	TOTAL GAIN(dB)	POT #
	[C]	[F]	[G]	[W]	[U]	[H]	[I]	[A]		
1 -	1	42	12	IN -- OUT	HYD HYD	42	---	54	--	1
2 -	2	42	18	IN -- OUT	HYD HYD	42	---	60	--	2
3 -	4	42	24	IN -- OUT	HYD HYD	42	---	66	--	3
4 -	3	42	30	IN -- OUT	HYD HYD	42	---	72	--	4
5 -	13	42	30	IN -- OUT	HYD HYD	42	---	72	--	5
6 -	19	42	30	IN -- OUT	HYD HYD	42	---	72	--	6
7 -	11	42	30	IN -- OUT	HYD HYD	42	---	72	--	7
8 -	12	42	30	IN -- OUT	HYD HYD	42	---	72	--	8
9 -	15	42	30	IN -- OUT	HYD HYD	42	---	72	--	9

```
>>> END OF LOG FILE =====
```

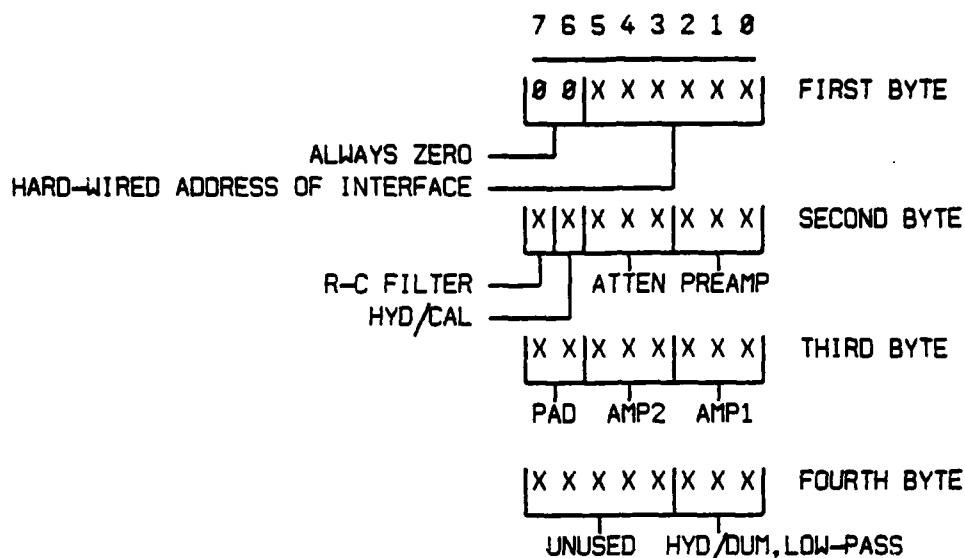
Figure 8. CHYDRA log-file and display format. Note that the letters in square brackets refer to commands in CHYDRA used to change values in the corresponding columns.

There are two labels shown in Figure 8 which can be quite confusing. The *POT #* label refers to the number in the sequence in which data is sent from the array to the ship. *POT #* is also referred to as *sequence number* and *logical number*. The *WIRED #* label refers to a particular set of electronics in one hydrophone pot. This wired number is the one that is always kept with the data, since the wired number refers to one specific hydrophone and related electronics.

5.3.3 Internal Format of Array Configuration

There is a much more compact internal format for the information contained in the display format of Figure 8. This internal format is stored on the minicomputer's operating-system disk to provide the previous array configuration upon starting CHYDRA. The configuration is stored in this format while being edited inside the CHYDRA program, and is transferred to the microprocessor and stored in the same format. Since the transfer takes a significant time over the 300 baud radio link to the array, a compact format is

Table 5
Internal format for the set-up
of one Hydra hydrophone interface



needed. The format shown in Table 5 is as compact as possible. Each interface currently has a maximum of 8 internal addresses containing 3 bits each (see Table 3 and Table 4), although only 19 of these 24 bits are presently used. Thus, the minimum number of bytes to fill these locations is 4, including the hard-wired address of the interface.

5.3.4 Interface Gain Control

The gains of the hydrophone amplifiers must be controlled in an optimum way, since there are 960 possible gain settings! Most of these gain settings are redundant, so a judicious choice of optimum gain configurations is made to obtain the lowest noise and to keep the number of gain combinations to a minimum. These optimum gain configurations are listed in Figure 9, which is the actual file CHYDRA.GAN that is read from by the program CHYDRA to determine the way in which it sets up the amplifier gain stages. You will note in Figure 8 that the preamp gain is specified separately from the rest of the gains. This is because the high-pass characteristics of the preamp depend on its gain. Therefore, both the preamp gain and another high-pass filter can be independently controlled to set the high-pass characteristics for each individual sensor.

The hexadecimal format of the gain codes in Figure 9 is chosen so that the third byte shown in Table 5 can be filled directly. Only the left column of hexadecimal numbers in Figure 9 are read by CHYDRA. The rest of the file is just a description of how the hexadecimal numbers are produced. The binary equivalents of the hexadecimal numbers are shown broken up into the bits used for the pad, gain 2 and gain 1. To the right of

7F	01 111 111	-36	0	0	-36	1
7E	01 111 110	-36	6	0	-30	2
7D	01 111 101	-36	12	0	-24	3
3F	00 111 111	-18	0	0	-18	4
3E	00 111 110	-18	6	0	-12	5
3D	00 111 101	-18	12	0	-6	6
BF	10 111 111	0	0	0	0	7
BE	10 111 110	0	6	0	6	8
BD	10 111 101	0	12	0	12	9
BC	10 111 100	0	18	0	18	10
B4	10 110 100	0	18	6	24	11
AC	10 101 100	0	18	12	30	12
A4	10 100 100	0	18	18	36	13
A3	10 100 011	0	24	18	42	14
9B	10 011 011	0	24	24	48	15
9A	10 011 010	0	30	24	54	16
92	10 010 010	0	30	30	60	17
91	10 010 001	0	36	30	66	18
89	10 001 001	0	36	36	72	19
88	10 001 000	0	42	36	78	20
80	10 000 000	0	42	42	84	21

HEX	PAD G2 G1	PAD	G1	G2	TOTAL	#
	BINARY	(dB)	(dB)	(dB)	(dB)	

THIS FILE IS AN OPTIMUM GAIN TABLE

PHILIP STAAL, 15:55pm Saturday, 15 November 1980

Figure 9. File of optimum gain configurations used for Hydra: CHYDRA.GAN

these, the individual gains corresponding to the binary numbers are shown. Finally, the totals of these gains are shown beside their row numbers.

5.4 Data Transfer to Related Software

The CHYDRA program is normally run before a data acquisition program. Since data acquisition programs store data from the Hydra array, they need information on the array configuration. This information is passed to them in the file CHYDRA.2DA which is on the operating-system disk.

The file CHYDRA.2DA is a one-block file which can be read into an integer array of 512 bytes such as LABEL(1 to 512). The format of this file is given in Figure 10. The number of hydrophone interfaces that are used is stored in LABEL(1). Each one of the interfaces in use is described with four bytes (such as LABEL(2) to LABEL(5) for interface 1). The first of these four bytes is a binary number representing the wired number of the interface to which the remaining three bytes apply. The second byte is a binary number representing the total gain (in decibels) of the interface. The third byte is

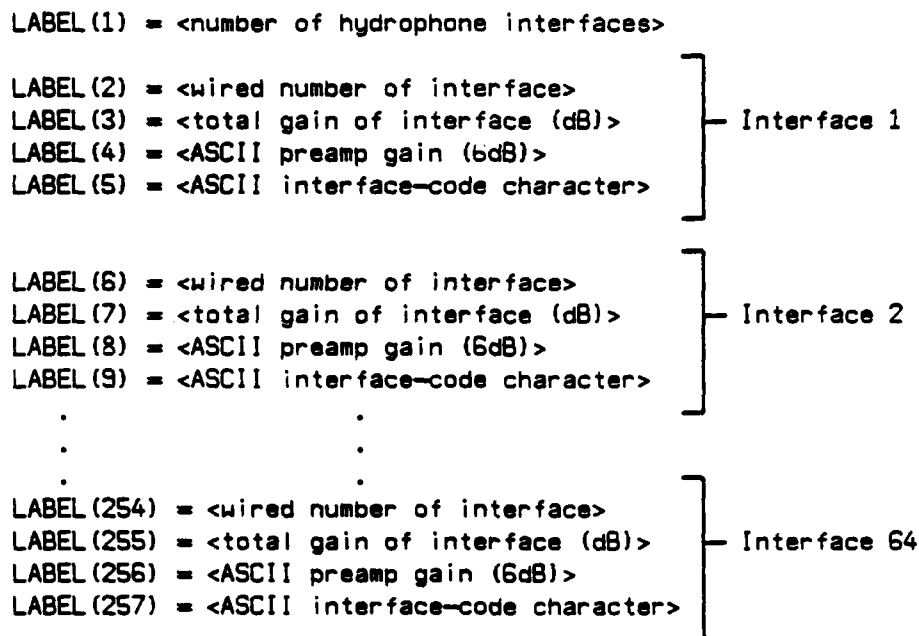


Figure 10. File format for data transfer to Hydra data acquisition software:
CHYDRA.2DA

an ASCII character in the range from "3" to "7" which represents the preamp gain. The preamp gain (in decibels) is determined by multiplying this character by 6. For example, if the <ASCII preamp gain (6dB)> is "7", then the preamp gain is $7 \times 6 = 42$ dB. The fourth byte is an ASCII character in the range from "A" to "H" or "Q" to "X". The conversion from this ASCII interface-code character to filter and hydrophone status is given in Table 6. For example, if the <ASCII interface-code character> is "A", then there is no calibration signal, the high-pass filter is out, the low-pass filter is in, and the input is from the hydrophone. The reason for this ASCII coding is that the current DREA-standard time-series file header has no locations defined for frequency response or calibration specifications.

Table 6

Conversion from ASCII character codes
to filter and hydrophone status in the Hydra array

ASCII INTERFACE- CODE CHARACTER	HYDROPHONE OR CAL	HIGH-PASS FILTER	LOW-PASS FILTER	HYDROPHONE OR DUMMY
A	HYD	OUT	IN	HYD
B	HYD	OUT	IN	DUM
C	HYD	OUT	OUT	HYD
D	HYD	OUT	OUT	DUM
E	HYD	IN	IN	HYD
F	HYD	IN	IN	DUM
G	HYD	IN	OUT	HYD
H	HYD	IN	OUT	DUM
Q	CAL	OUT	IN	HYD
R	CAL	OUT	IN	DUM
S	CAL	OUT	OUT	HYD
T	CAL	OUT	OUT	DUM
U	CAL	IN	IN	HYD
V	CAL	IN	IN	DUM
W	CAL	IN	OUT	HYD
X	CAL	IN	OUT	DUM

6 Concluding Remarks

The Hydra array system is very versatile and, as a result, very complex. Therefore, a relatively sophisticated software package is required to control the system, to display its configuration, and to log automatically the highly variable states of the system. Such a software package, described in this document, has been used to great advantage during the development of the Hydra array and during a multitude of different deployments over the past four years.

The Hydra array, controlled with the software described in this document, has been used to collect data reported in a number of papers, two of which are listed here: [Chapman and Ellis, 1982], and [Staal, 1983].

Having control software in the wet end of the array system as well as on the ship has proven to be advantageous. A simple terminal can be used to control all functions of the array without the rest of the shipborne computers. This enhances the reliability of the system, and enables technicians to work on the array without tying up a minicomputer. Also, the microprocessor in the wet end of the array system can do real-time changes to the array without intervention over the radio link by the shipborne minicomputer. Despite this split in the software package between the two processors, the minicomputer can have complete control, since it down-line-loads the software for the microprocessor.

Appendix A

Glossary

ASCII	American Standard Code for Information Interchange: Coded character set to be used for the general interchange of information among information-processing systems, communications systems, and associated equipment.
BAUD	A unit of telegraph signaling speed equal to the number of code elements (pulses and spaces) per second or twice the number of pulses per second.
BUS	One or more electrical conductors along which information and/or power is transmitted from any of several sources to any of several destinations.
COMMUTATOR	A device which combines several parallel data streams into a serial one.
CROSS-ASSEMBLER	A program which runs on one computer and converts symbolic computer code into machine language for a different type of computer.
DECOMMUTATOR	A device which separates a serial data stream into several parallel ones.
DMA	Direct Memory Access: The use of special hardware for direct transfer of data to or from memory to minimize the interruptions caused by program-controlled data transfers.
DOWNLOAD	To load a program stored in one computer into the memory of a second computer.
DUPLEX	The operation of associated transmitting and receiving apparatus concurrently, as in ordinary telephones, without manual switching between talking and listening periods.
FIFO	First-In, First-Out memory: Typically used for buffering data between asynchronous devices, or as a delay device.
FSK	Frequency Shift Keying: A method used for transferring digital information, in which different frequencies correspond to different discrete characters or symbols. Normally, one frequency corresponds to 1 and another frequency corresponds to 0.
H/P	HydroPhone: A device which receives underwater sound waves and converts them to electric waves.
POT	Hydrophone units including hydrophones, amplifiers, filters, interfaces etc. are loosely termed <i>pots</i> .

RAM	Random Access Memory: A data storage device having the property that the time required to access a randomly selected datum does not depend on the time of the last access or the location of the most recently accessed datum.
RF	Radio Frequency: A frequency at which coherent electromagnetic radiation of energy is useful for communication purposes; roughly the range from 10 kHz to 100 GHz.
ROM	Read Only Memory: A device for storing data in permanent, or non-erasable, form; usually a static electronic or magnetic device allowing extremely rapid access to data.
RS-232	An electrical and mechanical standard for connecting devices which send and receive information. Used to connect computers to computers, and computers to terminals, etc.
SHOT	An explosive underwater sound source.
SIMPLEX	A mode of radio transmission in which communication takes place between two stations in only one direction at a time.
SYNC WORD	Synchronization word: A digital word, chosen so that it is reliably detected, and used to mark the beginning of, or other known position in, a data sequence.
WET-END	The <i>wet-end</i> of the array system includes the array itself and directly connected electronics.

Appendix B

Instructions for Microprocessor Program UPCP4

In the following description, what is typed by the user is bold, and what is typed by the computer is in italics.

The UT4 utility program types a * when waiting for a command. Typing \$P0<CR>, where <CR> means Carriage-return, will start UPCP4.

The main routine types @ when waiting for one of the following commands:

UT4<CR>	Return to the UT4 utility program
LOW FREQ<CR>	Turn on the low-frequency calibration signal
RANDOM NOISE<CR>	Turn on the random-noise calibration signal
HALT CAL<CR>	Stop the calibration signal
BITS/WORD: <number><CR>	Set number of bits/word (6 to 12)
SEQ/SYNC: <number><CR>	Set number of sequences/sync (1 to 16)
# OF CHANNELS: <number><CR>	Set number of channels (1 to 63)
DIGITIZATION RATE: <number><CR>	Set digitization rate = 8192/<number> (Hz)
Z<CR>	Set up all H/P's from configuration table
POT #: <number><CR>	Set the logical interface number for one H/P.
	The program then types >
	and waits for one of the following commands:
WIRED #: <number><CR>	Set the interface address (base 16)
G0, DB: <number><CR>	Set preamp gain (18 to 42 dB)
G1, DB: <number><CR>	Set amplifier 1 gain (0 to 42 dB)
G2, DB: <number><CR>	Set amplifier 2 gain (0 to 42 dB)
ATTEN DB: <number><CR>	Set attenuator (0 to 42 dB)
	The pad may be 0, 18 or 36 dB:
HYD, H.P. FILTER (Y OR N) ? <Y or N>, PAD DB: <number><CR>	
C, H.P. FILTER (Y OR N) ? <Y or N>, PAD DB: <number><CR>	
DUMMY (Y OR N) ? <Y or N>, L.P. FILTER (Y OR N) ? <Y or N>	
	Anything else will cause a return to the @ prompt

Appendix C

Instructions for Minicomputer Program CHYDRA

The following set of instructions explain what is required of the operator to set up the Hydra array, assuming that the hardware and software are in place. In the following description, what is typed by the user is bold, and *what is typed by the computer is in italics*. <CR> means carriage-return.

First, start the CHYDRA program following the RT11 prompt:

.R CHYDRA<CR>

*? = List of available commands
Q = Quit back to monitor
A = Attenuator change
C = Configuration of array change
D = Direct connection of terminal to microprocessor
F = Frequency response (preamplifier gain) change
G = Gain of post amps change
H = Hydrophone or Cal change
I = Input hydrophone or capacitor change
L = Load microprocessor from PDP11 disk file
P = Print array configuration table
T = Type array configuration table
U = Upper frequencies (anti-alias filter) change
W = High-pass filter change
X = Transmit and setup array configuration*

COMMAND ?

You can now do any of the functions listed in this menu by typing the appropriate character followed by <CR>. To start the array from power-up, type **D<CR>**. The terminal will then appear to be directly connected to the microprocessor. The utility program should be started, using the commands listed in Table 1. The prompt ***** will be displayed when the utility program has started. Then, to get out of direct mode, type **<CTRL-C><CTRL-C>**. To load the program into the microprocessor, type **L<CR>** after the **COMMAND ?** prompt. The computer will then display *INPUT FILE NAME ? **. Type the file name **UPCP4<CR>** and the microprocessor will be loaded. The computer will then prompt you for another command. The number of bits per word, number of channels, number of sequences per sync and the digitization rate should be set by using command **D** and the instructions for the microprocessor program UPCP4 in Appendix B. Then the microprocessor should be left with the utility program running. To get out of direct mode, type **<CTRL-C><CTRL-C>**.

At this point, type **T<CR>** in response to the **COMMAND ?** prompt. You will see a display of the present array configuration similar to that in Figure 8. If you wish any changes to the configuration of the table, use the editing commands **A**, **C**, **F**, **G**, **H**, **I**, **U** or **W** to do the changes. When you are satisfied with the array configuration displayed, use

the command X to transmit the configuration and set up the array. Note that any hydrophone pots that have not been addressed in the displayed configuration will be turned off to save power. To turn all the hydrophone pots back on, you must power down the sub-surface electronics and re-power it using the D command and the commands in Table 1.

At any point, you may print the current array configuration table on the line printer with command P or type the table on your terminal with command T.

When you are through changing the array, *always stop the CHYDRA program by using command Q*. This is because the program must store the current configuration in CHYDRA.CFG for the next time CHYDRA is run. It must also store the configuration in CHYDRA.2DA for data acquisition programs and in CHYDRA.LOG for the continuing history of the array. If you stop the program with a <CTRL-C>, none of these things will be stored.

The CHYDRA.LOG file is defined to be 150 blocks, which is large enough to store roughly 50 array configurations. If the file becomes full while the program CHYDRA is storing an array configuration in it, the program tells you to rename the current CHYDRA.LOG file, to run CHYDRA again, and to quit CHYDRA in the normal fashion. The program will then create a new CHYDRA.LOG file, and store the last configuration in it.

Appendix D

Subroutines Used in Microprocessor Program UPCP4

CALLING SEQUENCE	NAME	FUNCTION PERFORMED
SEP 4 .WORD <subroutine address>	CALL	Calls a subroutine. This is used to implement all the other subroutine calls below.
SEP 5	RETURN	Returns from a subroutine.
SEP 4 .WORD BRNG .BYTE 'A' A 'B' B 0	BRNG	Gets one character from the console and branches to the address corresponding to the character. In this example, if A is typed the program branches to label A, if B is typed the program branches to label B, and if anything else is typed the program just continues.
SEP 4 .WORD CHKCR BR <address to go to if NOT carriage-return> <continues here if a carriage-return was received>	CHKCR	Gets one character from the console and branches accordingly.
SEP 4 .WORD CHK2 BR <address to go to if NOT carriage-return> <continues here if a carriage-return was received>	CHK2	Does the same as CHKCR except that the character to be checked is already in register RF.1
SEP 4 .WORD BEL	BEL	Types ?<BELL> to the console terminal.
SEP 4 .WORD NTRNO BR <error address> <continues here if a carriage-return was received>	NTRNO	Types : and accepts a two-digit decimal number from the console terminal. The hex equivalent is stored in register RD.1.
SEP 4 .WORD BLKTYP .WORD <address to start> .BYTE <number of characters>	BLKTYP	Types <number of characters> characters starting at <address to start> to the console terminal.
SEP 4 .WORD BUSOUT	BUSOUT	Outputs a number of commands down the bus to the hydrophone interfaces.

Appendix E

Subroutines Used In Minicomputer Program CHYDRA

E.1 Input/Output Subroutines

- QSTN** This subroutine types the list of available commands on the console terminal.
- QUIT(IM)** This subroutine returns to the monitor in a graceful fashion. QUIT stores the current configuration in CHYDRA.CFG, appends the configuration to CHYDRA.LOG, and outputs the configuration to CHYDRA.2DA for the data acquisition programs.
- DIRECT(ITER)** This subroutine connects the console terminal directly to terminal number ITER. To pass out of direct mode, type two control-C's in a row.
- SETTER(UNIT,OPEN)** This subroutine attaches the terminal UNIT and normally sets it for no echo, no wait, tab, form, no CR LF on carriage limit, handle XON/XOFF, no handle <control>-F,<control>-B,<control>-X, and pass all out if OPEN = 1. Pass all in is set if OPEN = 2. If OPEN = 0, then the terminal UNIT is reset to the initial characteristics and detached. If UNIT = 5, the FSK xmit is turned on and ASCII xmit is enabled for OPEN.NE.0, and the FSK xmit is turned off and data xmit is enabled for OPEN = 0. The baud rate is normally set to 300 baud. However, if UNIT = 4, then the baud rate is set to 9600 baud.
- DOWNLD(IM)** This subroutine asks for a single dump disk file, which is then downline loaded to the micro-processor. The program UCP4 is the most commonly used dump file.
- FILEIN(IFNAME,ISTART,IM,NCHAR)** This subroutine inputs and converts to LOAD format, the disk dump file specified by IFNAME, into the virtual array IM, starting at IM(ISTART). The number of elements read into IM is NCHAR. This subroutine is used by DOWNLD to get the dump file off disk.
- LOAD(ISTART,ISTOP,MSTART,IM)** This subroutine takes the memory image for the microprocessor stored from IM(ISTART) to IM(ISTOP), and downline loads it starting at the micro address specified by MSTART. This is used for loading programs and for sending array configuration information to the microprocessor.

ICLEAR	This function outputs anything remaining in the terminal 5 input buffer to the console terminal, then outputs up to 2 question marks to the micro to get the * prompt from UT4. If the response is OK, ICLEAR=1, if not, ICLEAR=0.
IC2UP(ICO)	This function outputs the character contained in ICO to the micro, and sets IC2UP=1 if the first returned character is the same, otherwise IC2UP=0. The input buffer must be empty before calling IC2UP.
OUTCFG(IM,IUNIT,IWRITE)	This subroutine types out the array configuration table to unit IUNIT. OUTCFG is used for output to the console terminal, the configuration display and to the printer. If IWRITE is 0, then IUNIT is the terminal number. If IWRITE is 1, then IUNIT refers to the FORTRAN unit number.
XMIT(IM)	This subroutine transmits the array configuration table to the microprocessor, and tells it to set up the array.

E.2 Configuration-editing Subroutines

CONFIG(IM)	This subroutine allows the reconfiguration of the active hydrophones by allowing the number active and their hardwired numbers to be entered.
HYDCAL(IM)	This subroutine allows the choice of Hydrophone alone, or Hydrophone in series with a calibration signal.
HIGHPS(IM)	This subroutine allows the choice of using a high-pass filter or not.
LOWPAS(IM)	This subroutine allows the choice of using an anti-alias filter or not.
DUMMYC(IM)	This subroutine allows the choice of the Hydrophone, or a dummy capacitor.
FREQ(IM)	This subroutine allows changing the preamp gains to 18,24,30,36, or 42 dB, which also affects the high-pass frequency characteristics.
ATTEN(IM)	This subroutine allows changing the attenuators to 0,6,12,18,24,30,36, or 42 dB.
POTRNG(IM,ISTART,IEND)	This subroutine inputs the range of pots to be changed. ISTART is the first and IEND is the last pot to be changed.

GAIN(IM)

This subroutine allows changing the post-amp gains and the pads to within the range of -36 to 84 dB.

E.3 Numerical-conversion Subroutines**H2I(IM,ISTART,INTEGR)**

This subroutine converts two ASCII hex digits starting at ISTART in IM into the integer number INTEGR.

I2H(IM,ISTART,INTEGR)

This subroutine converts the integer number INTEGR into two ASCII hex digits which are placed in IM starting at ISTART.

Appendix F

Microprocessor Array-control Program UPCP4

```

.PRINT
.PAGE
*
* CONTROL PROGRAM FOR HYDRA
* 10:37am Friday, 10 June 1983
*
* This version contains UPCP1 and UPCP3 plus additional
* commands to bring the program up to date with the new LEU, pots
* and receiver package. First to be used on cruise Q111.
*
* UT4 ENTRY POINTS
*
READ      =$813E      Input ascii into RF.1
READAH    =$813B      " and hex into RD from right, DF=1; else DF=0.
TYPE      =$81A4      Type ascii @RF.1
TYPE2     =$81AE      Type hex pair @RF.1
TYPE6     =$81A2      Type ascii immediate
*
STARTRH   =<STARTR>H
STARTRL   =<STARTR>L
STACKRH   =<STACK>H
STACKRL   =<STACK>L
CFIGH     =<CFIG>H
CFIGL     =<CFIG>L
CMANDRH   =<CMAND>H
CMANDRL   =<CMAND>L
CALLRH    =<CALL>H
CALLRL    =<CALL>L
RETRL     =<RETURN>L
UL        =<U>L
LL        =<L>L
RL        =<R>L
HL        =<H>L
BL        =<B>L
SL        =<S>L
NL        =<N>L
ZL        =<Z>L
DL        =<D>L
PL        =<P>L
WL        =<W>L
GL        =<G>L
AL        =<A>L
HYL       =<HY>L
CL        =<C>L
B0L       =<B0>L
B1L       =<B1>L
B2L       =<B2>L
DMYL      =<DMY>L
HPYL      =<HPY>L
HPNL      =<HPN>L
DMYYL     =<DMYY>L
DMYNL     =<DMYN>L
LPYL      =<LPY>L
LPNL      =<LPN>L
*=0
.SPACE
LDI       STARTRH Main program

```

```

PHI      3      starts at
LDI      STARTL START
PLO      3
LDI      STACKH High byte for
PHI      2      stack
LDI      STACKL Low byte for
PLO      2      stack bottom
LDI      CMANDH High byte for
PHI      8      command sequence
LDI      CFIGH  High byte for
PHI      $A     pot configuration
LDI      CALLH  High byte for
PHI      4      CALL
PHI      5      RETURN
LDI      CALLL  Low byte for
PLO      4      CALL
LDI      RETL   Low byte for
PLO      5      RETURN
START    SEP     3      Main program counter is R3
START    SEP     4      Type [CR] [LF] @ [SPACE]
        .WORD    BLKTYP  "
        .WORD    COMAT   "
        .BYTE    4       "
START    SEP     4      Branch subroutine
        .WORD    BRNG    "
        .BYTE    'U' UL  'L' LL 'R' RL 'H' HL
        .BYTE    'B' BL  'S' SL '#' NL 'P' PL
        .BYTE    'D' DL  'Z' ZL 0
U        BR      START  Error. None of above
        SEP     4      Type [T] [4]
        .WORD    BLKTYP  "
        .WORD    T4      "
        .BYTE    2       "
        SEP     4      Return to UT4
        .WORD    CHKCR   Check for [CR]
BR        START  Error
LDI      $80     Load R5 with 8039
PHI      5      start of UT4
LDI      $39     "
PLO      5      "
L        LBR     $812E  Branch to end of TIMALC
        SEP     4      Type 'ow Freq'
        .WORD    BLKTYP  "
        .WORD    LOWF    "
        .BYTE    7       "
        SEP     4      Low frequency
        .WORD    CHKCR   Check for [CR]
BR        START  Error
SEX      3      Output 5      0000/0101
OUT      2      to port 2
        .BYTE    5       "
R        BR      START  Start for more
        SEP     4      Type 'andom Noise'
        .WORD    BLKTYP  "
        .WORD    RAND    "
        .BYTE    11      "
        SEP     4      Random noise
        .WORD    CHKCR   Check for [CR]
BR        START  Error
SEX      3      Output 4      0000/0100
OUT      2      to port 2
        .BYTE    4       "

```

	BR	START	Start for more	
H	SEP	4	Type 'alt Cal'	
	.WORD	BLKTYP	"	
	.WORD	HALTC	"	
	.BYTE	7		
	SEP	4	Halt cal	
	.WORD	CHKCR	Check for [CR]	
	BR	START	Error	
	SEX	3	Output 2	0000/0010
	OUT	2	to port 2	
	.BYTE	2	"	
	BR	START	Start for more	
B	SEP	4	Type 'its/word'	
	.WORD	BLKTYP	"	
	.WORD	ITSWRD	"	
	.BYTE	8		
	SEP	4	Bits per word	
	.WORD	NTRNO	Enter # to RD.1	
	BR	START	Error	
	SEX	2	Stack pointer	
	GHI	\$D	Get #	
	STR	2	Store on stack	
	OUT	6	Output to port 6	
	DEC	2	Restore pointer	
	BR	START	Start for more	
S	SEP	4	Type 'eq/sync'	
	.WORD	BLKTYP	"	
	.WORD	EQSYNC	"	
	.BYTE	7		
	SEP	4	Sequences per sync	
	.WORD	NTRNO	Enter # to RD.1	
	BR	START	Error	
	SEX	2	Stack pointer	
	GHI	\$D	Get #	
	ORI	\$40	Bit rate = 131072 * 4	
	STR	2	Store on stack	
	OUT	7	Output to port 7	
	DEC	2	Restore pointer	
	BR	START	Start for more	
N	SEP	4	Type 'of channels'	
	.WORD	BLKTYP	"	
	.WORD	OFCHAN	"	
	.BYTE	12		
	SEP	4	# of channels	
	.WORD	NTRNO	Enter # to RD.1	
	BR	START	Error	
	SEX	2	Stack pointer	
	GHI	\$D	Get #	
	STR	2	Store on stack	
	OUT	5	Output to port 5	
	DEC	2	Restore pointer	
	OUT	1	Output to port 1	
	DEC	2	Restore pointer	
	BR	START	Start for more	
D	SEP	4	Type 'igitization rate'	
	.WORD	BLKTYP	"	
	.WORD	IGITIZ	"	
	.BYTE	16		
	SEP	4	Sampling rate = 8192/#	
	.WORD	NTRNO	Enter # to RD.1	
	BR	START	Error	
	SEX	2	Stack pointer	

	GHI	\$D	Get #
	STR	2	Store on stack
	OUT	3	Output to port 3
	DEC	2	Restore pointer
	BR	START	Start for more
P	LBR	POT	Branch to next page
Z	SEP	4	Pot setup
	.WORD	CHKCR	Check for [CR]
	BR	START	Error
	LDI	CFIGL	Set RA to point at #of pots
	PLO	\$A	in configuration table
	LDI	0	Zero R9.0
	PLO	9	"
COUT	INC	9	R9=R9+1
	INC	\$A	Set RA at wired address
	LDI	CMANDH	Point R8 at
	PHI	8	storage
	LDI	CMANDL	bottom
	PLO	8	for command sequence
	SEX	8	Get pot wired
	LDN	\$A	address
	PLO	7	into R7.0
	ORI	\$C0	Add control bits for reset Q
	STXD		Store on command stack
	INC	\$A	Point at H/C, atten, G0 byte
	LDN	\$A	Load byte
	PHI	7	Put byte in R7.1
	ANI	\$38	Mask atten
	ORI	2	Address of atten = 2
	STXD		Store on command stack
	LDN	\$A	Reload byte
	ANI	7	Mask G0
	SHL		Shift left
	SHL		to put preamp gain
	SHL		in data location
	ORI	4	Address of preamp = 4
	STXD		Store on command stack
	INC	\$A	Point at pad, G2, G1 byte
	LDN	\$A	Load pad, G2, G1 byte
	ANI	\$C0	Mask pad
	SHR		Shift pad attenuation
	SHR		into data
	SHR		position
	STR	8	Store on stack
	GHI	7	Get H/C, Atten, G0 byte
	ANI	\$40	Mask H/C bit
	SHR		Shift to data position
	OR		Add pad bits
	STR	8	Store on stack
	GHI	7	Get H/C, Atten, G0 byte
	SHLC		Put filter bit into DF
	BDF	FILTIN	If DF = 1, filter in
	LDI	6	Address for filter out
	LSKP		Skip past FILTIN
FILTIN	LDI	3	Address for filter in
	OR		Add filter bit
	STXD		Store on command stack
	LDN	\$A	Reload pad, G2, G1 byte
	ANI	\$38	Mask G2
	ORI	1	Address of G2 = 1
	STXD		Store on command stack
	LDN	\$A	Reload pad, G2, G1 byte

	ANI	7	Mask G1
	SHL		Shift left
	SHL		to put gain
	SHL		in data location
	STXD		Address of G1=0, store on stack
	INC	\$A	Point at HYD or DUM, L.P. byte
	LDN	\$A	Load byte
	ANI	7	Mask HYD or DUM, L.P.
	SHL		Shift left
	SHL		to put HYD or DUM, L.P.
	SHL		in data location
	ORI	5	Address of HYD or DUM, L.P. = 5
	STXD		Store on command stack
	GLO	7	Get wired address
	ORI	\$80	Add bits for set Q
	STXD		Store on command stack
	GLO	9	Get logical pot #
	STXD		Store on command stack
	GLO	7	Get wired address
	ORI	\$40	Set address bit
	STXD		Store on command stack
	LDI	10	Ten commands
	STR	8	Store on command stack
	SEP	4	Output commands
	.WORD	BUSOUT	to bus
	LDI	CFIGH	Set R8
	PHI	8	to point
	LDI	CFIGL	at #
	PLO	8	of pots
	GLO	9	Get logical pot #
	SEX	8	R8 is pointer
	SD		#of pots-logical pot#
	LBNZ	COUT	Go back for more pots
	LBR	START	Start for more
POT	SEP	4	Type 'ot #'
	.WORD	BLKTYP	"
	.WORD	POTNUM	"
	.BYTE	4	
	SEP	4	Pot #
	.WORD	NTRNO	Enter # to RD.1
	LBR	START	Error
	GHI	\$D	Get #
	PLO	9	Put in R9.0
	SHL		*2
	SHL		*2
	SMI	3	-3 to get offset in CFGI table
	ADI	CFIGL	Add base address of array config.
	PLO	\$A	RA is present pot pointer
MOREP	SEP	4	Type [CR] [LF] > [SPACE]
	.WORD	BLKTYP	"
	.WORD	ARROW	"
	.BYTE	4	"
	LDI	CMANDL	Point R8 at storage bottom
	PLO	8	for command sequence
	SEX	8	Get pot wired
	LDN	\$A	address
	ORI	\$C0	Add control bits for reset Q
	STXD		Store on command stack
	SEP	4	Branch subroutine
	.WORD	BRNG	"
	.BYTE	'W' WL 'G' GL 'A' AL 'H' HYL	
	.BYTE	'C' CL 'D' DMYL 0	

	LBR	START	Error
W	SEP	4	Type 'ired #'
	.WORD	BLKTYP	"
	.WORD	WIREDN	"
	.BYTE	6	
	SEP	4	Wired #
	.WORD	NTRNO	Enter #
	BR	MOREP	Error
	SEX	8	Command pointer
	GLO	9	Logical pot #
	STXD		Store on command stack
	GLO	\$D	Get wired #
	STR	\$A	Store in array config. memory
	ORI	\$40	Add control bits for set address
	STXD		Store on command stack
	LDI	2	Two commands
ENDW	STR	8	Store on command stack
	SEP	4	Output commands
	.WORD	BUSOUT	to bus
	BR	MOREP	Branch for more
G	SEP	4	Branch subroutine
	.WORD	BRNG	"
	.BYTE	'0' B0L	'1' B1L '2' B2L 0
	BR	MOREP	Error
B0	LDI	4	Set preamp address
	LSKP		
B1	LDI	0	Set amp 1 address
	LSKP		
B2	LDI	1	Set amp 2 address
	STR	8	Store on command stack
	SEP	\$C	Delay of
	.BYTE	23	3 bit times
	SEP	4	Type a
	.WORD	TYPE6	comma
	.BYTE	','	"
DECBEL	SEP	4	Type ' DB'
	.WORD	BLKTYP	"
	.WORD	DB	"
	.BYTE	3	"
	SEP	4	Enter #
	.WORD	NTRNO	"
	BR	MOREP	Error
	LDI	7	Put 7 into R0.0
	PLO	0	"
	GHI	\$D	Get #
DBLP	BZ	ZERO	Go to ZERO if #=zero
	DEC	0	Increment R0
	SMI	6	Subtract 6 dB
	NOP		
	NOP		
	BNF	ERR	Error if # not multiple of 6
	BR	DBLP	Go to subtract 6 more
ERR	SEP	\$C	Delay of
	.BYTE	23	3 bit times
	SEP	4	Type [LF]
	.WORD	TYPE6	"
	.BYTE	\$A	"
	SEP	4	Type error
	.WORD	BEL	message
	BR	MOREP	Jump to more pot commands
ZERO	GLO	0	Get # of 6's that were subtracted
	SHL		Put data in correct

	SHL		location in command
	SHL		"
	SEX	8	Command pointer
	OR		Add address bits
	STXD		Store on command stack
FINI	LDN	\$A	Get wired number
	ORI	\$80	Add bits to make set Q
	STXD		Store on command stack
	LDI	3	Three commands
	BR	ENDW	Go back to output commands
A	SEP	4	Type 'tten'
	.WORD	BLKTYP	"
	.WORD	TTEN	"
	.BYTE	4	
	LDI	2	Set attenuator address
	STR	8	Store on command stack
	BR	DECBEL	Go to enter dB
C	LBR	PG2C	Branch to next page
DMY	LBR	PG2DMY	"
HY	SEP	4	Type 'yd'
	.WORD	BLKTYP	"
	.WORD	YD	"
	.BYTE	2	
	LDI	3	Set cal/pad address
SAME	STR	2	Store on general stack
	DEC	2	Decrement stack pointer
	SEP	4	Type ', H.P. filter (Y or N) ? '
	.WORD	BLKTYP	"
	.WORD	HPFILT	"
	.BYTE	25	
	SEP	4	Branch subroutine
	.WORD	BRNG	"
	.BYTE	'Y' HPYL 'N' HPNL 0	
	BR	SAME	Error. None of above
HPN	LDI	3	Adjust address for no filter
	LSKP		
HPY	LDI	0	
	SEX	2	Point at general stack
	IRX		Point at cal/pad address
	ADD		Adjust if necessary
	STR	8	Store on command stack
	SEP	4	Type ', pad dB'
	.WORD	BLKTYP	"
	.WORD	PADDB	"
	.BYTE	8	
	SEP	4	Enter #
	.WORD	NTRNO	"
	LBR	MOREP	Error
	SEX	8	Command pointer
	GLO	\$D	Get #
	BZ	C0	Go to C0 if 0
	XRI	\$18	Go to C18
	BZ	C18	if 18
	GLO	\$D	Get #
	XRI	\$36	Go to C36
	BZ	C36	if 36
	LBR	ERR	Error
C18	DEC	8	Leave data 0 if 18 dB
	LBR	FINI	Go to complete command
C0	LDI	\$10	Data 10 if 0 dB
	LSKP		
C36	LDI	8	Data 8 if 36 dB

	OR		Add address bits
	STXD		Store on command stack
	LBR	FINI	Go to complete command
PG2C	LDI	\$23	Set cal bits
	BR	SAME	Go to complete command
PG2DMY	SEP	4	Type 'ummy (Y or N) ? '
	.WORD	BLKTYP	"
	.WORD	UMMY	"
	.BYTE	16	"
	SEP	4	Branch subroutine
	.WORD	BRNG	"
	.BYTE	'Y' DMYYL 'N' DMYNL 0	
	LBR	DMY	Error.None of above
DMYY	LDI	\$D	Set dummy H/P bit and address
	LSKP		
DMYN	LDI	5	Address for data
	STR	2	Put on general stack
	DEC	2	Set stack pointer
	SEP	4	Type ', L.P. filter (Y or N) ? '
	.WORD	BLKTYP	"
	.WORD	LPFILT	"
	.BYTE	25	"
	SEP	4	Branch subroutine
	.WORD	BRNG	"
	.BYTE	'Y' LPYL 'N' LPNL 0	
	BR	DMYN	Error.None of above
LPN	LDI	\$10	Set anti-alias filter bypass bit
	LSKP		
LPY	LDI	0	
	SEX	2	Use general stack pointer
	IRX		Point at dummy and address info
	OR		Combine info
	SEX	8	Point with command stack pointer
	STXD		Store on command stack
	LBR	FINI	Go to complete command

*

*

*

Pot configuration table

CFIG	.BYTE	0	#of logical pots
	.BYTE	0 0 0 0	pot #1
	.BYTE	0 0 0 0	pot #2
	.BYTE	0 0 0 0	pot #3
	.BYTE	0 0 0 0	pot #4
	.BYTE	0 0 0 0	pot #5
	.BYTE	0 0 0 0	pot #6
	.BYTE	0 0 0 0	pot #7
	.BYTE	0 0 0 0	pot #8
	.BYTE	0 0 0 0	pot #9
	.BYTE	0 0 0 0	pot #10
	.BYTE	0 0 0 0	pot #11
	.BYTE	0 0 0 0	pot #12
	.BYTE	0 0 0 0	pot #13
	.BYTE	0 0 0 0	pot #14
	.BYTE	0 0 0 0	pot #15
	.BYTE	0 0 0 0	pot #16
	.BYTE	0 0 0 0	pot #17
	.BYTE	0 0 0 0	pot #18
	.BYTE	0 0 0 0	pot #19
	.BYTE	0 0 0 0	pot #20
	.BYTE	0 0 0 0	pot #21
	.BYTE	0 0 0 0	pot #22
	.BYTE	0 0 0 0	pot #23

```

        .BYTE 0 0 0 0 pot #24
        .BYTE 0 0 0 0 pot #25
        .BYTE 0 0 0 0 pot #26
        .BYTE 0 0 0 0 pot #27
        .BYTE 0 0 0 0 pot #28
        .BYTE 0 0 0 0 pot #29
        .BYTE 0 0 0 0 pot #30
        .BYTE 0 0 0 0 pot #31
        .BYTE 0 0 0 0 pot #32
*
*      Command buffer
*
        .BYTE 0 0 0 0 0
        .BYTE 0      10 commands
        .BYTE 0      wired address
        .BYTE 0      logical #
        .BYTE 0      set Q
        .BYTE 0      hyd/dum, L. P.
        .BYTE 0      G1
        .BYTE 0      G2
        .BYTE 0      cal/pad
        .BYTE 0      G0
        .BYTE 0      atten
CMAND   .BYTE 0      reset Q
*
*      Stack
*
        .BYTE 0 0 0 0 0 0 0 0
        .BYTE 0 0 0 0 0 0 0
STACK   .BYTE 0
*
*      VOCABULARY
*
COMAT   .BYTE $0D $0A 'e '
T4      .BYTE 'T4'
LOWF    .BYTE 'ow Freq'
RAND    .BYTE 'andom'
        .BYTE 'Noise'
HALTC   .BYTE 'alt Cal'
ITSWRD  .BYTE 'its/word'
EQSYNC  .BYTE 'eq/sync'
OFCHAN  .BYTE ' of chan'
        .BYTE 'nels'
IGITIZ  .BYTE 'igitizat'
        .BYTE 'ion rate'
POTNUM  .BYTE 'ot #'
ARROW   .BYTE $0D $0A '> '
COLON   .BYTE ':'
DB      .BYTE ' DB'
QSTN    .BYTE '? ' $07
WIREDN  .BYTE 'ired #'
TTEN    .BYTE 'tten'
YD      .BYTE 'yd'
PADDB   .BYTE ', pad dB'
HPFILT  .BYTE ', H.P. f'
        .BYTE 'ilter (Y'
        .BYTE ' or N) ?'
        .BYTE ' '
UMMY    .BYTE 'ummy (Y '
        .BYTE 'or N) ? '
LPFILT  .BYTE ', L.P. f'
        .BYTE 'ilter (Y'

```

```

        .BYTE    ' or N) ?'
        .BYTE    ' '
        .SPACE

*
*   SUBROUTINES FOR CP1
*   9:55am Wednesday, 27 April 1983
*
*   Branching subroutine
*
BRNG    SEP      4      Enter ascii to RF.1
        .WORD    READAH  "
        SEX      6      R6 points at subroutine parameters
MTCHLP  GHI      $F      Get ascii
        XOR      6      Compare with parameter
        BZ       MATCH   Branch if match
        LDA      6      Pick up next parameter
        INC      6
        BNZ      MTCHLP  Loop if not 0
        SEP      4      Type error
        .WORD    BEL     "
        DEC      6
        SEP      5      Error return
MATCH   INC      6      Pick up next parameter
        LDA      6      for branch
        PLO      6      "
        SEP      5      Return

*
*   Check [CR] subroutine
*
CHKCR   GHI      $E      Get RE.1
        ORI      1      Set bit one for echo off
        PHI      $E      Replace
        SEP      4      Enter ascii
        .WORD    READAH  to RF.1
        GHI      $E      Get RE.1
        ANI      $FE     Reset bit one for echo on
        PHI      $E      Replace
CHK2    GHI      $F      Get ascii
        XRI      $D      XOR with [CR]
        BNZ      BEL     Branch to BEL if not [CR]
        INC      6      Skip error jump
        INC      6      in main routine
        SEP      5      Return
BEL     SEP      4      Type '? [BEL] '
        .WORD    BLKTYP  "
        .WORD    QSTN    "
        .BYTE    2      "
        SEP      5      Return

*
*   Enter number subroutine
*
*   RD.1 is the binary equivalent of the input 0 to 99 base 10.
*
NTRNO   SEP      4      Type': '
        .WORD    BLKTYP  "
        .WORD    COLON   "
        .BYTE    2      "
        LDI      0      Put 0
        PLO      $D      in RD.0
HEXLP   SEP      4      Read hex
        .WORD    READAH  into RD
        LBDF     HEXLP   Branch if hex read

```

```

BNZ      BUSLP      not finished
SEX      3          Program pointer
OUT      4          Leave bus
.BYTE    $FF        FF to port 4
SEP      $C        Delay of
.BYTE    1          0.4 bit times
DEC      8          Return command pointer to bottom
SEP      5          Return

*
*      Call subroutine
*
CLLP      SEP      3      Go to subroutine
CALL      SEX      2      Point to stack
          GHI      6      R6 to stack to prepare
          STXD     for pointing to
          GLO      6      arguments and decrement
          STXD     to free location
          GHI      3      R3 to R6
          PHI      6      to save
          GLO      3      return address
          PLO      6      "
          LDA      6      Load address of
          PHI      3      subroutine
          LDA      6      into R3
          PLO      3      "
          BR       CLLP    Reset call pointer

*
*      Return subroutine
*
RTNLP      SEP      3      Return to main program
RETURN     GHI      6      R6 to R3
          PHI      3      R3 contains
          GLO      6      the return address
          PLO      3      "
          SEX      2      Point to stack
          INC      2      Point to saved old R6
          LDXA     Restore contents
          PLO      6      of R6
          LDX      "
          PHI      6      "
          BR       RTNLP    Reset return pointer
          .SPACE
          .SPACE
          .END

```

Appendix G

Minicomputer Array-control Program CHYDRA

C-----DREA-SWA-STAA-----

PROGRAM CHYDRA

Written by Philip Staal

LAST EDIT: 3:44pm Monday, 16 May 1983

PURPOSE: This program controls the HYDRA array through the terminal 5 connection, and lists the current array configuration on terminal 4. It can print the Hydra configuration and appends each configuration upon Quitting the program into CHYDRA.LOG.

HARDWARE: - terminal 5 connection to HYDRA array
- terminal 4 connection to video terminal

SOFTWARE: - gain file SY:CHYDRA.GAN
- previous configuration file SY:CHYDRA.CFG
- micro-processor files to be downloaded
- file SY:CHYDRA.2DA
- append file SY:CHYDRA.LOG

CONFIGURATION STORAGE FORMAT	BYTE #	PURPOSE
	1	# of logical #'s
	2	Pot 1 wired #
	3	Pot 1 L. F. Rolloff, H/C, atten, freq (preamp)
	4	Pot 1 gain (pad, G1, G2)
	5	Pot 1 hyd/dum, L. P. filt.
	6	Pot 2 wired #
	7	Pot 2 L. F. ...
	8	Pot 2 gain ...
	:	:
	:	:
	:	:

COMMON VIRTUAL ARRAY IM(32000)

IM(1) to IM(29999) uP Dump Programs

IM(31000), IM(31001)	'0', '0' to '3', 'F'	# of logical #'s
IM(31002), IM(31003)	'0', '0' to '3', 'F'	wired #'s
IM(31004), IM(31005)	'0', '0' to 'F', 'F'	HP, H/C, atten, freq
IM(31006), IM(31007)	'0', '0' to 'F', 'F'	gain
IM(31008), IM(31009)	'0', '0' to 'F', 'F'	hyd/dum, L. P. filt.
:	:	:
:	:	:

IM(31500), IM(31501)	-36 dB in ASCII hex form
IM(31502), IM(31503)	-30 dB "
:	:
:	:

IM(31540), IM(31541)	80 dB "
----------------------	---------

```

C      IM(31600) = 1    If changes made to configuration table
C      = 0            that have not been transmitted
C      = 0            Otherwise
C
C      NOTE: For 300 BAUD, <LF> takes 15.4 ticks, and any other
C      character takes 2.2 ticks. Sleep times are taken from
C      this, with a 5 tick safety margin.
C
C      VIRTUAL IM(32000)
C      COMMON/TERS/SBLOK          ! terminal status block
C      BYTE IM
C      IM(31600)=0                ! No changes to xmit
C
C      Read in optimum gain table, and last array configuration
C
C      OPEN(UNIT=2, NAME='SY: CHYDRA. GAN', READONLY, TYPE='OLD')
C      DO 10 I=1, 41, 2
10      READ(2, 10020) IM(31499+I), IM(31500+I)
C      CLOSE(UNIT=2)
C      OPEN(UNIT=2, NAME='SY: CHYDRA. CFG', TYPE='OLD')
C      READ(2, 10020) IM(31000), IM(31001)
C      CALL H2I(IM, 31000, NP) ! Get number of pots in config.: NP
C      DO 20 J=1, NP
C      L=31002+(J-1)*8
20      READ(2, 10030) IM(L), IM(L+1), IM(L+2), IM(L+3), IM(L+4), IM(L+5), IM(L+
16), IM(L+7)
C      CLOSE(UNIT=2)
C
C      Type command list and look for commands
C
C      CALL QSTN
30      TYPE 10040
C      ACCEPT 10050, CMD
C      IF (CMD. EQ. 'A') CALL ATTEN(IM)
C      IF (CMD. EQ. 'C') CALL CONFIG(IM)
C      IF (CMD. EQ. 'D') CALL DIRECT(5)
C      IF (CMD. EQ. 'F') CALL FREQ(IM)
C      IF (CMD. EQ. 'G') CALL GAIN(IM)
C      IF (CMD. EQ. 'H') CALL HYDCAL(IM)
C      IF (CMD. EQ. 'I') CALL DUMMYC
C      IF (CMD. EQ. 'L') CALL DOWNLD(IM)
C      IF (CMD. EQ. 'P') CALL OUTCFG(IM, 6, 1)
C      IF (CMD. EQ. 'Q') CALL QUIT(IM)
C      IF (CMD. EQ. 'T') CALL OUTCFG(IM, 0, 0)
C      IF (CMD. EQ. 'U') CALL LOWPAS(IM)
C      IF (CMD. EQ. 'W') CALL HIGHPS(IM)
C      IF (CMD. EQ. 'X') CALL XMIT(IM)
C      IF (CMD. EQ. 'A'. OR. CMD. EQ. 'C'. OR. CMD. EQ. 'D') GO TO 30
C      IF (CMD. EQ. 'F'. OR. CMD. EQ. 'G'. OR. CMD. EQ. 'H') GO TO 30
C      IF (CMD. EQ. 'I'. OR. CMD. EQ. 'L'. OR. CMD. EQ. 'P') GO TO 30
C      IF (CMD. EQ. 'Q'. OR. CMD. EQ. 'T'. OR. CMD. EQ. 'U') GO TO 30
C      IF (CMD. EQ. 'W'. OR. CMD. EQ. 'X') GO TO 30
C
C      Type menu if command is not recognized
C
C      CALL QSTN
C      GO TO 30
C
C      10010  FORMAT(I4)
C      10020  FORMAT(2A1)
C      10030  FORMAT(8A1)
C      10040  FORMAT(/4X, 'COMMAND ? '$)

```


10050 FORMAT (A1)
END

C-----DREA-SWA-STAAAL-----
SUBROUTINE QSTN

C
C This subroutine types the list of parameters and
C available commands on the console terminal.

TYPE 10010
RETURN

C
10010 FORMAT(/4X,'? = List of available commands'/
1 4X,'Q = Quit back to monitor'/
2 4X,'A = Attenuator change'/
3 4X,'C = Configuration of array change'/
4 4X,'D = Direct connection of terminal to microprocessor'/
5 4X,'F = Frequency response (preamp gain) change'/
6 4X,'G = Gain of post amps change'/
7 4X,'H = Hydrophone or Cal change'/
8 4X,'I = Input hydrophone or capacitor change'/
9 4X,'L = Load microprocessor from PDP11 disk file'/
1 4X,'P = Print array configuration table'/
2 4X,'T = Type array configuration table'/
3 4X,'U = Upper frequencies (anti-alias filter) change'/
4 4X,'W = High-pass filter change'/
5 4X,'X = Transmit and setup array configuration')
END

C-----DREA-SWA-STAAAL-----
SUBROUTINE QUIT(IM)

C
C This subroutine returns to the monitor in a graceful
C fashion. It stores the current configuration in CHYDRA.CFG,
C appends the configuration to CHYDRA.LOG, and outputs the
C information to CHYDRA.2DA for the data acquisition programs.

VIRTUAL IM(32000)
BYTE IM, LABEL(512), CHAR
INTEGER*2 DBLK(4)
DATA LABEL/512*0/ ! data for CHYDRA.2DA
DATA DBLK/3RSY, 3RCHY, 3RDRA, 3R2DA/ ! SY:CHYDRA.2DA is out file

C
IF(IM(31600).EQ.1)TYPE 10020 ! Warn if config. not xmitted

C
C Ask if really want to quit

TYPE 10030
ACCEPT 10040, YESNO
IF(YESNO.NE.'Y')RETURN

C
C Store last array config. in SY:CHYDRA.CFG, and in SY:CHYDRA.2DA

TYPE 10110
OPEN(UNIT=2, NAME='SY:CHYDRA.CFG', TYPE='UNKNOWN',
1CARRIAGECONTROL='LIST')
ICHAN=IGETC
IF(ICHAN.LT.0)STOP '*** NO CHANNEL IN QUIT ***'
IF(IFETCH(DBLK).NE.0)STOP '*** BAD FETCH IN QUIT ***'
IF(IENTER(ICHAN,DBLK,1).LT.0)STOP '*** BAD IENTER IN QUIT ***'
WRITE(2,10050)IM(31000),IM(31001) ! write hex # of pots
CALL H2I(IM,31000,NP)

C
C Loop over all pots

```

C
10  LABEL(1)=NP                      ! Number of pots
    DO 40 I=1,NP
        IPOT=4*I-2
        L=31002+(I-1)*8
        CALL H2I(IM,L,NW)           ! wired number
        CALL H2I(IM,L+2,NHC)       ! HiPass, H/C, atten, preamp
        CALL H2I(IM,L+4,NG)       ! gain (pad, G1, G2)
        CALL H2I(IM,L+6,NDUM)     ! hyd/dum, L. P. filt.
        DO 20 J=2,42,2
            CALL H2I(IM,31498+J,NTST)
20      IF(NG.EQ.NTST)GO TO 30
30      NG=J*3-42                   ! Post amp and pad gain
        IGPRE=(7-(NHC.AND."7"))*6 ! Preamp gain
        NGT=NG+IGPRE               ! Total gain
        LABEL(IPOT)=NW            ! Wired number
        LABEL(IPOT+1)=NGT
        LABEL(IPOT+2)=IGPRE/6+48  ! Pre-amp gain / 6 (dB)
        IPCODE=65+(NDUM.AND."3")+(NHC.AND."200")/32+(NHC.AND."100")/4
        LABEL(IPOT+3)=IPCODE      ! ASCII pot-code
40      WRITE(2,10060)IM(L),IM(L+1),IM(L+2),IM(L+3),IM(L+4),IM(L+5),
        1IM(L+6),IM(L+7)          ! write to CHYDRA.CFG
        ICODE=IWRITE(256,LABEL,0,ICHAN) ! write to CHYDRA.2DA
        IF(ICODE.LT.0)STOP '*** BAD IWRITE IN QUIT ***'
        CALL CLOSEC(ICHAN)
        CALL IFREEC(ICHAN)
        CLOSE(UNIT=2)

C
C
C      Append the configuration to SY:CHYDRA.LOG.

C
C
C      TYPE 10100
50      OPEN(UNIT=2,NAME='SY:CHYDRA.LOG',TYPE='OLD',
        1CARRIAGECONTROL='FORTRAN',ERR=60)
        GO TO 80

C
C
C      If no file exists, create one of NBLOKS blocks

C
C
C      NBLOKS=150
60      TYPE 10080,NBLOKS/10
        OPEN(UNIT=2,NAME='SY:CHYDRA.LOG',TYPE='NEW',
        1CARRIAGECONTROL='FORTRAN',INITIALSIZE=NBLOKS)
        WRITE(2,10070)
        DO 70 I=1,NBLOKS-1
70      WRITE(2,10090)             ! blank out the file so RT11 keeps its size
        CLOSE(UNIT=2)
        GO TO 50

C
C
C      Find end of file

C
C
C      80      READ(2,10040,END=100,ERR=100)CHAR
        IF(CHAR.EQ.'>')GO TO 90
        GO TO 80

C
C
C      Output configuration

C
C
C      90      BACKSPACE 2
100     CALL OUTCFG(IM,2,1)
        WRITE(2,10070,ERR=110,END=110)
        CLOSE(UNIT=2)
        STOP 'End of program CHYDRA'
110     STOP 'CHYDRA.LOG full.  Rename it,
        1 then RUN CHYDRA and Quit again.'

```

```

C
10010  FORMAT(I4)
10020  FORMAT(/4X,'<<< Configuration Table Changes Not Transmitted
1 >>>')
10030  FORMAT(/4X,'STOP --- ARE YOU SURE ? (Y/N) '$)
10040  FORMAT(A1)
10050  FORMAT(2A1)
10060  FORMAT(8A1)
C
C      End of log file marker
C
10070  FORMAT(1X/1X/' >>> END OF LOG FILE =====')
C
C      Informative messages and blank block
C
10080  FORMAT(/4X,'Creating CHYDRA.LOG file.  Wait',I3,' seconds.'/)
10090  FORMAT(1X,126X,3(/1X,126X))
10100  FORMAT(/4X,'Appending configuration to CHYDRA.LOG file.'/)
10110  FORMAT(/4X,'Storing configuration in CHYDRA.CFG and CHYDRA.2DA')
      END
C-----DREA-SWA-STAA-----
      SUBROUTINE DIRECT(ITER)
C
C      This subroutine connects the console terminal directly to
C      ITER. To pass out of direct mode, type two control-C's in a row.
C
      BYTE ICHAR, ICPO
C
      IFLAG=0
      ICPO=0          ! previous output character
C
      Set up terminal characteristics
C
      CALL SCCA(IFLAG)      ! stop intercepting control-C
      CALL SETTER(0, 2)
      CALL SETTER(ITER, 2)
C
      Send characters from console to iter
C
10      IGOT1=MTIN(0, ICHAR)
      IF(IGOT1.NE.0)GO TO 30 ! if no console character, check iter
      IF(ICHAR.EQ.3.AND.ICPO.EQ.3)GO TO 50 ! quit if double control-C
20      ITST=MTOUT(ITER, ICHAR)
      IF(ITST.EQ.1)GO TO 20
      ICPO=ICHAR
C
      Send characters from iter to console
C
30      IGOT1=MTIN(ITER, ICHAR)
      IF(IGOT1.NE.0)GO TO 10 ! if no iter character, check console
40      ITST=MTOUT(0, ICHAR)
      IF(ITST.EQ.1)GO TO 40
      GO TO 30
C
      Reset terminal characteristics
C
50      CALL SCCA
      CALL SETTER(0, 0)
      CALL SETTER(ITER, 0)
      RETURN
      END
C-----DREA-SWA-STAA-----

```

SUBROUTINE SETTER(UNIT, OPEN)

This subroutine attaches the terminal UNIT and normally sets it for no echo, no wait, tab, form, no CR LF on carriage limit, handle XON/XOFF, no handle ↑F, ↑B, ↑X, and pass all out if OPEN = 1. Pass all in is set if OPEN = 2. If OPEN = 0, then the terminal UNIT is reset to the initial characteristics and detached. If UNIT = 5, the FSK xmit is turned on and ASCII xmit is enabled for OPEN.NE.0, and the FSK xmit is turned off and data xmit is enabled for OPEN = 0. The baud rate is normally set to 300 baud. However, if UNIT = 4, then the baud rate is set to 9600 baud.

INTEGER*2 SBLOK(4, 6, 2), UNIT, OPEN

IZERO=0

IF(OPEN.EQ.0) GO TO 20

Set the following on OPEN.NE.0

IER=MTATCH(UNIT, 0, JOB)

IF(IER.NE.0) STOP '*** MTATCH ERROR IN SETTER ***'

IER=MTGET(UNIT, SBLOK(1, UNIT+1, 1))

IF(IER.NE.0) STOP '*** MTGET ERROR IN SETTER ***'

DO 10 I=1, 4 ! save the terminal characteristics

10 SBLOK(I, UNIT+1, 2)=SBLOK(I, UNIT+1, 1)

SBLOK(1, UNIT+1, 1)="152705

SBLOK(2, UNIT+1, 1)="100006

IF(UNIT.EQ.4) SBLOK(1, UNIT+1, 1)="157305 ! set for display

IF(OPEN.EQ.2) SBLOK(2, UNIT+1, 1)="100206 ! read pass all too

IER=MTSET(UNIT, SBLOK(1, UNIT+1, 1))

IF(IER.NE.0) STOP '*** MTSET ERROR IN SETTER ***'

CALL RCTRLO

IF(UNIT.NE.5) RETURN

C CALL IPOKE("176510, "4.OR. IPEEK("176510)) ! Used for TT2:

CALL IPOKE("160114, "2000.OR. IPEEK("160114)) ! Turn on FSK xmit

ISEC=1

ITICK=5

CALL ISLEEP(IZERO, IZERO, ISEC, IZERO) ! Wait for xmitter

IT=MTOUT(UNIT, 22) ! Output ↑V for xmit ASCII

CALL ISLEEP(IZERO, IZERO, IZERO, ITICK) ! Wait for switch

RETURN

Reset the following if OPEN = 0

20 IF(UNIT.NE.5) GO TO 30

IT=MTOUT(UNIT, 20) ! Output ↑T for xmit data

ITICK=5

CALL ISLEEP(IZERO, IZERO, IZERO, ITICK)

C CALL IPOKE("176510, .NOT. "4.AND. IPEEK("176510))! Turn off FSK xmit

CALL IPOKE("160114, .NOT. "2000.AND. IPEEK("160114))

30 DO 40 I=1, 4 ! restore the terminal characteristics

40 SBLOK(I, UNIT+1, 1)=SBLOK(I, UNIT+1, 2)

IER=MTSET(UNIT, SBLOK(1, UNIT+1, 1))

IF(IER.NE.0) STOP '*** MTSET ERROR IN SETTER ***'

IER=MTDTCH(UNIT)

IF(IER.NE.0) STOP '*** MTDTC ERROR IN SETTER ***'

RETURN

END

C-----DREA-SWA-STAA-----

SUBROUTINE DOWNLD(IM)

C

```

C          This subroutine asks for a single dump disk file, which
C          is then downline loaded to the micro-processor.
C
VIRTUAL IM(32000)
BYTE IM,MSTART(7)
INTEGER*2 IFSPEC(39)          ! input file specifications
REAL*4 EXT(2)
DATA EXT/6RDMPDAT,6RDATDAT/    ! DMP is default file type

C          Get the input file name
C
TYPE 10010
IF(ICS1(IFSPEC,EXT,,0).NE.0)GO TO 10

C          Input the file to array IM
C
CALL FILEIN(IFSPEC,1,IM,NCHAR)

C          NCHAR characters of dump data have been input.
C          Now put the up loading address in MSTART
C
DO 20 I=1,6
  MSTART(I)=IM(I)
MSTART(7)=' '

C          Open the link to the micro, transmit the dump data, close link
C
CALL SETTER(5,1)
CALL LOAD(7,NCHAR,MSTART,IM)
CALL SETTER(5,0)
RETURN

C
10010  FORMAT(/4X,'INPUT FILE NAME ? '$)
END

C-----DREA-SWA-STAA-----
SUBROUTINE FILEIN(IFNAME,ISTART,IM,NCHAR)

C          This subroutine inputs and converts to LOAD format, the
C          disk dump file specified by IFNAME, into the virtual array IM,
C          starting at IM(ISTART). The number of elements read into IM
C          is NCHAR. This subroutine is used by DOWNLD to get the dump
C          file off disk.
C
VIRTUAL IM(32000)
BYTE IM,LINE(80)
INTEGER*2 IFNAME(39)

C          Open the input file
C
CALL IASIGN(2,IFNAME(16),IFNAME(17),0,32)
I=ISTART

C          Read and translate line by line to end of file,
C          ignoring blanks and skipping memory line-headers.
C
10  READ(2,10010,END=40)(LINE(J),J=1,80)
  K=6
  IF(I.EQ.ISTART)K=1
20  IF(LINE(K).EQ.' ')GO TO 30
  IF(LINE(K).EQ.';')GO TO 10
  IM(I)=LINE(K)
  I=I+1

```

```

30      K=K+1
      IF (K.GT.80) GO TO 50
      GO TO 20
C
C      Close file and return
C
40      NCHAR=I-1
      CLOSE (UNIT=2)
      RETURN
50      TYPE *, '*** ERROR READING FILE IN FILEIN ***'
      I=NSTART
      GO TO 40
C
10010  FORMAT (80A1)
      END
C-----DREA-SWA-STAA-----
      SUBROUTINE LOAD (ISTART, ISTOP, MSTART, IM)
C
C      This subroutine takes the memory image for the
C      microprocessor stored from IM(ISTART) to IM(ISTOP), and downline
C      loads it starting at the micro address specified by MSTART.
C
      VIRTUAL IM(32000)
      BYTE IM, MSTART(7), ICHAR
C
      IZERO=0
C
C      Get micro's prompt character *
C
      IF (ICLEAR().NE.1) RETURN
C
C      Tell micro where to put code
C
      DO 10 I=1,7
10      IF (IC2UP (MSTART (I)).NE.1) RETURN
C
C      Send code to micro, then <CR>
C
      DO 20 J=ISTART, ISTOP
20      IF (IC2UP (IM (J)).NE.1) RETURN
      IF (IC2UP (13).NE.1) RETURN
C
C      Wait 26 ticks, then check for error or prompt
C
      ITICK=26
      CALL ISLEEP (IZERO, IZERO, IZERO, ITICK)
30      IGOT1=MTIN (5, ICHAR)
      IF (ICCHAR.EQ.'?') GO TO 40
      IF (IGOT1.EQ.0) GO TO 30
      IF (ICCHAR.EQ.'*') RETURN
      TYPE *, '*** NO RESPONSE FROM UT4 IN LOAD ***'
      RETURN
C
40      TYPE *, '*** UNRECOGNIZED BY UT4 IN LOAD ***'
      RETURN
      END
C-----DREA-SWA-STAA-----
      FUNCTION ICLEAR
C
C      This function outputs anything remaining in the terminal
C      5 input buffer to the console terminal, then outputs up to 2
C      question marks to the micro to get the * prompt from UT4. If the

```

```

C      response is OK, ICLEAR=1, if not, ICLEAR=0.
C
C      BYTE ICHAR, ICPI
C
C      IZERO=0
C      ICLEAR=1
C      ICPI=0                ! previous input character
C      CALL RCTRLO          ! reset the control-0 terminal command
C
C      Output anything remaining in input buffer to the console
C
C      10      IGOT1=MTIN(5, ICHAR)
C              IF (IGOT1.NE.0) GO TO 40
C              IF (ICAR.NE.10) GO TO 20
C              IF (ICPI.EQ.13) GO TO 30
C      20      ITST=MTOUT(0, ICHAR)                ! display on the console
C              IF (ITST.EQ.1) GO TO 20
C      30      ICPI=ICAR
C              GO TO 10
C
C      Output up to 2 question marks and look for prompt
C
C      40      ITICK=48
C              DO 70 I=1, 2
C      50          ITST=MTOUT(5, '?')
C              IF (ITST.EQ.1) GO TO 50
C              CALL ISLEEP (IZERO, IZERO, IZERO, ITICK)
C      60          IGOT1=MTIN(5, ICHAR)
C              IF (IGOT1.EQ.0) GO TO 60
C      70          IF (ICAR.EQ. '*') RETURN
C              TYPE *, '*** NO RESPONSE FROM UT4 IN ICLEAR ***'
C              ICLEAR=0
C              RETURN
C              END
C-----DREA-SWA-STAA-----
C      FUNCTION IC2UP(ICO)
C
C          This function outputs the character contained in ICO to
C          the micro, and sets IC2UP=1 if the first returned character is
C          the same, otherwise IC2UP=0. The input buffer must be empty
C          before calling IC2UP.
C
C      BYTE ICO, ICI
C
C      Output character ICO
C
C      IC2UP=0
C      10      ITST=MTOUT(5, ICO)
C              IF (ITST.EQ.1) GO TO 10
C
C      Check up to 10000 times for a response
C
C      DO 20 I=1, 10000
C          IGOT1=MTIN(5, ICI)
C      20      IF (IGOT1.EQ.0) GO TO 30
C
C      If no response, say so and return
C
C      TYPE *, '*** NO RESPONSE FROM MICRO IN IC2UP ***'
C      RETURN
C
C      Check to see if the response matches what was sent

```

```

C
30  IF (ICI.NE.ICO) GO TO 40
    IC2UP=1
    RETURN

C
C    If response doesn't match, say so and give the ADE numbers
C
40  TYPE *, '*** MICRO ECHOED WRONG CHARACTER IN IC2UP ***'
    IICI=ICI
    IICO=ICO
    TYPE 10010, IICI, IICO
    RETURN

C
10010  FORMAT(9X, 'Sent: ', I3, ', Returned: ', I3)
      END

C-----DREA-SWA-STAA-----
      SUBROUTINE H2I (IM, ISTART, INTEGR)

C
C          This subroutine converts two ASCII hex digits starting
C          at ISTART in IM into the integer number INTEGR.
C
      VIRTUAL IM(32000)
      BYTE IM

C
      N1=IM(ISTART)-48
      N2=IM(ISTART+1)-48
      IF (N1.GT.9) N1=N1-7
      IF (N2.GT.9) N2=N2-7
      INTEGR=N1*16+N2
      RETURN
      END

C-----DREA-SWA-STAA-----
      SUBROUTINE I2H (IM, ISTART, INTEGR)

C
C          This subroutine converts the integer number INTEGR into
C          two ASCII hex digits which are placed in IM starting at ISTART.
C
      VIRTUAL IM(32000)
      BYTE IM

C
      N1=INTEGR/16
      N2=INTEGR-(N1*16)
      IF (N1.GT.9) N1=N1+7
      IF (N2.GT.9) N2=N2+7
      IM(ISTART)=N1+48
      IM(ISTART+1)=N2+48
      RETURN
      END

C-----DREA-SWA-STAA-----
      SUBROUTINE CONFIG (IM)

C
C          This subroutine allows the reconfiguration of the active
C          hydrophones by allowing the number active and their hardwired
C          numbers to be entered.
C
      VIRTUAL IM(32000)
      BYTE IM

C
      IM(31600)=1                ! Changes to xmit

C
C          Get number of pots and put in configuration storage
C

```



```

10      TYPE 10010
        ACCEPT *, NP
        IF (NP. LT. 1. OR. NP. GT. 64) GO TO 10
        CALL I2H(IM, 31000, NP)
C
C      Get wired numbers of pots and put in configuration storage
C
        DO 30 I=1, NP
20      TYPE 10020, I
        ACCEPT *, NW
        IF (NW. LT. 1. OR. NW. GT. 64) GO TO 20
        LOCN=31002+(I-1)*8
30      CALL I2H(IM, LOCN, NW)
C
C      Type the revised array configuration on the console
C
        CALL OUTCFG(IM, 0, 0)
        RETURN
C
10010   FORMAT(/4X, 'NUMBER OF POTS ? '$)
10020   FORMAT(/4X, 'POT #', I3, ': WIRED # ? '$)
        END
C-----DREA-SWA-STAA-----
SUBROUTINE HYDCAL(IM)
C
C      This subroutine allows the choice of Hydrophone alone,
C      or Hydrophone in series with a calibration signal.
C
        VIRTUAL IM(32000)
        BYTE IM, IH
C
C      Get the sequence numbers of the pots to be changed
C
        CALL POTRNG(IM, ISTART, IEND)
C
C      Ask if hydrophone or cal required
C
10      TYPE 10010
        ACCEPT 10020, IH
        IF (IH. NE. 'H'. AND. IH. NE. 'C') GO TO 10
C
C      Put the hyd/cal information in configuration storage
C
        DO 20 I=ISTART, IEND
            LOCN=31004+(I-1)*8
            CALL H2I(IM, LOCN, NHC)
            IF (IH. EQ. 'H') NHC=. NOT. "100. AND. NHC
            IF (IH. EQ. 'C') NHC="100. OR. NHC
20      CALL I2H(IM, LOCN, NHC)
C
C      Type the revised array configuration on the console
C
        CALL OUTCFG(IM, 0, 0)
        RETURN
C
10010   FORMAT(/4X, 'HYDROPHONE (H) OR CAL (C) ? '$)
10020   FORMAT(A1)
        END
C-----DREA-SWA-STAA-----
SUBROUTINE HIGHPS(IM)
C
C      This subroutine allows the choice of having a high-pass

```

```

C      filter.
C
C      VIRTUAL IM(32000)
C      BYTE IM, IH
C
C      Get the sequence numbers of the pots to be changed
C
C      CALL POTRNG(IM, ISTART, IEND)
C
C      Ask if high-pass filter required
C
C      TYPE 10010
10     ACCEPT 10020, IH
      IF (IH. NE. 'I'. AND. IH. NE. 'O') GO TO 10
C
C      Put the high-pass information in configuration storage
C
C      DO 20 I=ISTART, IEND
C         LOCN=31004+ (I-1)*8
C         CALL H2I (IM, LOCN, NHC)
C         IF (IH. EQ. 'O') NHC=. NOT. "200. AND. NHC
C         IF (IH. EQ. 'I') NHC="200. OR. NHC
20     CALL I2H (IM, LOCN, NHC)
C
C      Type the revised array configuration on the console
C
C      CALL OUTCFG (IM, 0, 0)
C      RETURN
C
C      10010  FORMAT (/4X, 'HIGHPASS IN (I) OR OUT (O) ? '$)
10020  FORMAT (A1)
      END

```

-----DREA-SWA-STAA-----

```

C      SUBROUTINE LOWPAS (IM)
C
C          This subroutine allows the choice of having an
C          anti-alias filter.
C
C      VIRTUAL IM(32000)
C      BYTE IM, IL
C
C      Get the sequence numbers of the pots to be changed
C
C      CALL POTRNG (IM, ISTART, IEND)
C
C      Ask if anti-alias filter required
C
C      TYPE 10010
10     ACCEPT 10020, IL
      IF (IL. NE. 'I'. AND. IL. NE. 'O') GO TO 10
C
C      Put the anti-alias information in configuration storage
C
C      DO 20 I=ISTART, IEND
C         LOCN=31008+ (I-1)*8
C         CALL H2I (IM, LOCN, NDUM)
C         IF (IL. EQ. 'I') NDUM=. NOT. "2. AND. NDUM
C         IF (IL. EQ. 'O') NDUM="2. OR. NDUM
20     NDUM="3. AND. NDUM
C         CALL I2H (IM, LOCN, NDUM)
C
C      Type the revised array configuration on the console

```

```

C      CALL OUTCFG(IM, 0, 0)
      RETURN
C
10010  FORMAT(/4X, 'ANTI-ALIAS FILTER IN (I) OR OUT (O) ? '$)
10020  FORMAT(A1)
      END
C-----DREA-SWA-STAA-----
      SUBROUTINE DUMMYC(IM)
C
C      This subroutine allows the choice of the Hydrophone,
C      or a dummy capacitor.
C
      VIRTUAL IM(32000)
      BYTE IM, ID
C
C      Get the sequence numbers of the pots to be changed
C
      CALL POTRNG(IM, ISTART, IEND)
C
C      Ask if hydrophone or dummy required
C
10     TYPE 10010
      ACCEPT 10020, ID
      IF(ID.NE. 'H'.AND. ID.NE. 'D')GO TO 10
C
C      Put the hyd/dum information in configuration storage
C
      DO 20 I=ISTART, IEND
        LOCN=31008+(I-1)*8
        CALL H2I(IM, LOCN, NDUM)
        IF(ID.EQ. 'H')NDUM=.NOT. "1.AND. NDUM
        IF(ID.EQ. 'D')NDUM="1.OR. NDUM
        NDUM="3.AND. NDUM
        CALL I2H(IM, LOCN, NDUM)
20
C
C      Type the revised array configuration on the console
C
      CALL OUTCFG(IM, 0, 0)
      RETURN
C
10010  FORMAT(/4X, 'HYDROPHONE (H) OR DUMMY (D) ? '$)
10020  FORMAT(A1)
      END
C-----DREA-SWA-STAA-----
      SUBROUTINE FREQ(IM)
C
C      This subroutine allows changing the preamp gains
C      to 18, 24, 30, 36, or 42 dB, which also affects the high-pass
C      characteristics.
C
      VIRTUAL IM(32000)
      BYTE IM
C
C      Get the sequence numbers of the pots to be changed
C
      CALL POTRNG(IM, ISTART, IEND)
C
C      Ask for the preamp gain
C
10     TYPE 10010
      ACCEPT *, NF

```

```

C
C
C
IF (NF-NF/6*6.NE.0)GO TO 10      ! Must be multiple of 6
NF=NF/6
IF (NF.LT.3.OR.NF.GT.7)GO TO 10  ! Must be 18 to 42 dB

Put the preamp gain in configuration storage

NF=7-NF
DO 20 I=ISTART,IEND
  LOCN=31004+(I-1)*8
  CALL H2I(IM,LOCN,NHC)
  NHC=(*170.AND.NHC).OR.NF
  CALL I2H(IM,LOCN,NHC)

Type the revised array configuration on the console

CALL OUTCFG(IM,0,0)
RETURN

C
10010  FORMAT(/4X,'PREAMP GAIN (dB): (18,24,30,36,42) ? '$)
END
C-----DREA-SWA-STAA-----
SUBROUTINE ATTEN(IM)

      This subroutine allows changing the attenuators
to 0,6,12,18,24,30,36, or 42 dB.

VIRTUAL IM(32000)
BYTE IM

Get the sequence numbers of the pots to be changed

CALL POTRNG(IM,ISTART,IEND)

Ask for the attenuation

TYPE 10010
ACCEPT *,NAT
IF (NAT-NAT/6*6.NE.0)GO TO 10      ! Must be multiple of 6
IF (NAT.LT.0.OR.NAT.GT.42)GO TO 10 ! Must be between 0 and 42 dB

Put the attenuation in configuration storage

NAT=(7-NAT/6)*8
DO 20 I=ISTART,IEND
  LOCN=31004+(I-1)*8
  CALL H2I(IM,LOCN,NHC)
  NHC=(*107.AND.NHC).OR.NAT
  CALL I2H(IM,LOCN,NHC)

Type the revised array configuration on the console

CALL OUTCFG(IM,0,0)
RETURN

C
10010  FORMAT(/4X,'ATTENUATOR (dB): (0,6,12,18,24,30,36,42) ? '$)
END
C-----DREA-SWA-STAA-----
SUBROUTINE POTRNG(IM,ISTART,IEND)

      This subroutine inputs the range of pots to be changed.
ISTART is the first and IEND is the last pot to be changed.

```

VIRTUAL IM(32000)
 BYTE IM

C

IM(31600)=1 ! Changes to xmit
 CALL H2I(IM, 31000, NP)
 TYPE 10010
 ACCEPT *, ISTART, IEND
 IF(ISTART.LT.1.OR.ISTART.GT.IEND) GO TO 10
 IF(IEND.GT.NP) GO TO 10
 RETURN

C

10010 FORMAT(/4X, 'START POT, END POT ? '\$)
 END

C

-----DREA-SWA-STAA-----

SUBROUTINE GAIN(IM)

C

C

This subroutine allows changing the post-amp gains and
 the pads to within the range of -36 to 84 dB.

C

VIRTUAL IM(32000)
 BYTE IM

C

C

Get the sequence numbers of the pots to be changed

C

CALL POTRNG(IM, ISTART, IEND)

C

C

Ask for the gain excluding the preamp

C

C

10

TYPE 10010
 ACCEPT *, NG
 NG=NG+36
 IF(NG-NG/6*6.NE.0) GO TO 10 ! Must be multiple of 6
 IF(NG.LT.0.OR.NG.GT.120) GO TO 10 ! must be between -36 and 84 dB

C

C

Put the gain in configuration storage

C

NG=NG/3
 NG1=IM(31500+NG)
 NG2=IM(31501+NG)
 DO 20 I=ISTART, IEND
 LOCN=31006+(I-1)*8
 IM(LOCN)=NG1
 IM(LOCN+1)=NG2

20

C

C

Type the revised array configuration on the console

C

CALL OUTCFG(IM, 0, 0)
 RETURN

C

10010

FORMAT(/4X, 'GAIN (dB): '/' (-36, -30, -24, -18, -12, -6, 0, 6, 12, 18, 24,
 130, 36, 42, 48, 54, 60, 66, 72, 78, 84) ? '\$)
 END

C

-----DREA-SWA-STAA-----

SUBROUTINE OUTCFG(IM, IUNIT, IWRITE)

C

C

This subroutine types out the array configuration table
 to unit IUNIT. If IWRITE is 0, then IUNIT is the terminal
 number. If IWRITE is 1, then IUNIT refers to the FORTRAN unit
 number.

C

VIRTUAL IM(32000)
 BYTE IM, LINE(80), BLANK(2)

```

C      REAL*4 HC, HD, HIO, LIO
C      DATA BLANK/' ', 0/
C      IZERO=0
C      Read the time from the time code generator for output
C      CALL READTC(IID, IIH, IIM, IIS, IIMS)
C      IHZ=IIH
C      IMZ=IIM
C      ISZ=IIS
C      IMSZ=IIMS
C      CALL DAYMON(IID, IIH, IIM, IIS, IIMS)
C      Find the number of pots
C      CALL H2I(IM, 31000, NP)
C      Jump if FORTRAN write
C      IF(IWRITE.EQ.1)GO TO 30
C      Otherwise use multi-terminal output
C      CALL SETTER(IUNIT, 1)
C      LEN=79
C      IF(IUNIT.NE.4)GO TO 20
C      Clear screen for terminal 4
C      ICHAR=12
10    ITST=MTOUT(4, ICHAR)
C      IF(ITST.EQ.1)GO TO 10
C      ITICK=1      !wait 1 tick for clear
C      CALL ISLEEP (IZERO, IZERO, IZERO, ITICK)
C      Write blank line
C      CALL SCOPY (BLANK, LINE)
20    CALL MTPRNT (IUNIT, LINE)
C      Write date and time line
C      ENCODE (LEN, 10030, LINE) IIH/3+1, IIM, IHZ, IMZ, ISZ, IMSZ
C      CALL MTPRNT (IUNIT, LINE)
C      Write blank line
C      CALL SCOPY (BLANK, LINE)
C      CALL MTPRNT (IUNIT, LINE)
C      Write headings
C      ENCODE (LEN, 10040, LINE)
C      CALL MTPRNT (IUNIT, LINE)
C      ENCODE (LEN, 10050, LINE)
C      CALL MTPRNT (IUNIT, LINE)
C      ENCODE (LEN, 10060, LINE)
C      CALL MTPRNT (IUNIT, LINE)
C      Write blank line

```

```

C
CALL SCOPY(BLANK, LINE)
CALL MTPRNT(IUNIT, LINE)
GO TO 40      ! Go to start of loop

C
C
C
C
C
30  IF(IUNIT.NE.6)WRITE(IUNIT,10030,END=80,ERR=80)
    1      IIH/3+1, IIM, IHZ, IMZ, ISZ, IMSZ
    IF(IUNIT.EQ.6)WRITE(IUNIT,10080) IIH/3+1, IIM, IHZ, IMZ, ISZ, IMSZ

C
C
C
C
C
    Write blank line
    WRITE(IUNIT,10010,END=80,ERR=80)

C
C
C
C
    Write headings
    IF(IUNIT.NE.6)WRITE(IUNIT,10040,END=80,ERR=80)
    IF(IUNIT.EQ.6)WRITE(IUNIT,10090)
    IF(IUNIT.NE.6)WRITE(IUNIT,10050,END=80,ERR=80)
    IF(IUNIT.EQ.6)WRITE(IUNIT,10100)
    IF(IUNIT.NE.6)WRITE(IUNIT,10060,END=80,ERR=80)
    IF(IUNIT.EQ.6)WRITE(IUNIT,10110)

C
C
C
C
C
    Write blank line
    WRITE(IUNIT,10010,END=80,ERR=80)

C
C
C
C
    Convert and output NP configurations
40  DO 70 I=1,NP
    LOCN=31002+(I-1)*8
    CALL H2I(IM, LOCN, NW)
    CALL H2I(IM, LOCN+2, NHC)
    CALL H2I(IM, LOCN+4, NG)
    CALL H2I(IM, LOCN+6, NDUM)
    DO 50 J=2,42,2
    CALL H2I(IM, 31498+J, NTST)
50  IF(NG.EQ.NTST)GO TO 60
60  NG=J*3-42
    IGPRE=(7-(NHC.AND."7"))*6
    IATT=(7-(NHC.AND."70")/8)*6
    IHC=(NHC.AND."100")/64
    IHIO=(NHC.AND."200")/128
    IHD=(NDUM.AND."1")
    ILIO=(NDUM.AND."2")/2
    HC='HYD'
    IF(IHC.EQ.1)HC='CAL'
    HIO='OUT'
    IF(IHIO.EQ.1)HIO=' IN'
    HD='HYD'
    IF(IHD.EQ.1)HD='DUM'
    LIO=' IN'
    IF(ILIO.EQ.1)LIO='OUT'
    NGT=NG+IGPRE
    IF(IWRITE.EQ.0)ENCODE(LEN,10070,LINE)
    1      I, NW, IGPRE, NG, HIO, LIO, HC, HD, IATT, NGT, I
    IF(IWRITE.EQ.0)CALL MTPRNT(IUNIT, LINE)
    IF((IWRITE.EQ.1).AND.(IUNIT.NE.6))WRITE(IUNIT,10070,END=80,
    1      ERR=80) I, NW, IGPRE, NG, HIO, LIO, HC, HD, IATT, NGT, I
70  IF((IWRITE.EQ.1).AND.(IUNIT.EQ.6))WRITE(IUNIT,10120)

```

```

1          I, NW, IGP, NG, HIO, LIO, HC, HD, IATT, NGT, I
IF(IWRITE.EQ.0) CALL SETTER(IUNIT,0)
IF(IWRITE.EQ.0) RETURN

C
C
C      Write formfeed for printer

IF(IUNIT.EQ.6) WRITE(IUNIT,10020)
IF(IUNIT.NE.6) RETURN

C
C
C      Force printer to output buffer

CLOSE(UNIT=IUNIT)
OPEN(UNIT=IUNIT)
RETURN

C
C
C      Error on write

80      STOP 'CHYDRA.LOG full.  Rename it,
1 then RUN CHYDRA and Quit again.'

C
10010    FORMAT(1X)          ! blank line for write
10020    FORMAT('1')        ! formfeed for write
10030    FORMAT(' DATE: 83-',I2,'-',I2,' ', TIME:',I3,':',
112,':',I2,'.',I3,' ', CHYDRA version 83-5-16',5X)
10040    FORMAT(' POT WIRED PRE-AMP PAD,G1,G2 HIPASS AALIAS HYD HYD
1 ATTEN TOTAL POT')
10050    FORMAT(' # # GAIN(dB) GAIN(dB) FILTER FILTER CAL DUM
1 (dB) GAIN(dB) # ')
10060    FORMAT(' [C] [F] [G] [W] [U] [H] [I]
1 [A]',14X)
10070    FORMAT(1X,I2,' <-',I3,' <---',I3,' <-----',I4,' <----- ',A3,' <- ',
1 A3,2X,A3,1X,A3,I4,' <---',I4,' <-',I3)

C
C
C      Duplicate formats spaced over 11 spaces for printer

10080    FORMAT(////12X,'DATE: 83-',I2,'-',I2,' ', TIME:',I3,':',
112,':',I2,'.',I3,' ', CHYDRA version 83-5-16')
10090    FORMAT(12X,'POT WIRED PRE-AMP PAD,G1,G2 HIPASS AALIAS HYD HYD
1 ATTEN TOTAL POT')
10100    FORMAT(12X,' # # GAIN(dB) GAIN(dB) FILTER FILTER CAL DUM
1 (dB) GAIN(dB) # ')
10110    FORMAT(12X,' [C] [F] [G] [W] [U] [H] [I]
1 [A]')
10120    FORMAT(12X,I2,' <-',I3,' <---',I3,' <-----',I4,' <----- ',A3,' <- ',
1 A3,2X,A3,1X,A3,I4,' <---',I4,' <-',I3)
END

C-----DREA-SWA-STAA-----
SUBROUTINE XMIT(IM)

C
C      This subroutine transmits the array configuration table
C      to the microprocessor, and tells it to set up the array.
C
VIRTUAL IM(32000)
BYTE IM,MSTART(7),ICHAR,ICHAR1,ICHAR2

C
DATA MSTART/'1','M','0','2','8','0',' '/ 1M0280

C
IZERO=0
CALL H2I(IM,31000,NP) ! find the number of pots
CALL SETTER(5,1) ! open the link to the array

C
C      Load Hydra config. table into uProcessor, then check that

```



```

C      UT4 is still running.
C
      CALL LOAD(31000, 31001+NP*8, MSTART, IM)
      IF(ICLEAR 0.NE.1)GO TO 90
C
C      Type $P0 <CR> to start control program, wait 43 ticks,
C      then error if second last returned character is not @.
C
      IF(IC2UP('$').NE.1)GO TO 90
      IF(IC2UP('P').NE.1)GO TO 90
      IF(IC2UP('0').NE.1)GO TO 90
      IF(IC2UP(13).NE.1)GO TO 90
      ITICK=43
      CALL ISLEEP(IZERO, IZERO, IZERO, ITICK)
      ICHAR1=0
10      IGOT1=MTIN(5, ICHAR)
      IF(IGOT1.EQ.1)GO TO 20
      ICHAR2=ICHAR1
      ICHAR1=ICHAR
      GO TO 10
20      IF(ICHAR2.EQ.'@')GO TO 30
      TYPE *, '*** CONTROL PROGRAM FAILED TO START IN XMIT ***'
      GO TO 90
C
C      Type Z <CR> to uProcessor to set up array, wait 30 ticks,
C      then error if a ? is returned or if second last returned
C      character is not @.
C
30      IF(IC2UP('Z').NE.1)GO TO 90
      IF(IC2UP(13).NE.1)GO TO 90
      ITICK=30
      CALL ISLEEP(IZERO, IZERO, IZERO, ITICK)
40      IGOT1=MTIN(5, ICHAR)
      IF(IGOT1.EQ.1)GO TO 50
      IF(ICHAR.EQ.'?')GO TO 80
      ICHAR2=ICHAR1
      ICHAR1=ICHAR
      GO TO 40
50      IF(ICHAR2.EQ.'@')GO TO 60
      TYPE *, '*** CONTROL PROGRAM QUIT IN XMIT ***'
      GO TO 90
C
C      Type U to uProcessor, wait 10 ticks, type <CR>
C      to get uP back to UT4 monitor
C
60      IF(IC2UP('U').NE.1)GO TO 90
      ITICK=10
      CALL ISLEEP(IZERO, IZERO, IZERO, ITICK)
70      IGOT1=MTIN(5, ICHAR)
      IF(IGOT1.EQ.0)GO TO 70
      IT=IC2UP(13)
      IM(31600)=0
      CALL OUTCFG(IM, 4, 0)
      GO TO 90
                                     ! No changes to xmit
                                     ! Output config. to TT4:
C
C      Type error if Z command doesn't work
C
80      TYPE *, '*** Z UNRECOGNIZED BY CONTROL PROGRAM IN XMIT ***'
90      CALL SETTER(5, 0)
      RETURN
      RETURN
C-----DREA-SWA-STAA-----
      END

```

References

[Staal, Hughes and Olsen, 1981]

Staal, P.R., R.C. Hughes and J.H. Olsen. (1981). "Modular Digital Hydrophone Array". *Proceedings of IEEE Oceans '81*, Boston, September 1981, pp. 518 - 521. Re-issued as DREA TM 82/C.

[Chapman and Ellis, 1982]

Chapman D.M.F. and Dale D. Ellis. (1982). "Geo-acoustic Models for Propagation Modelling in Shallow Water". *Can. Acoust.* 11(2),9-24 (1983); also presented as an invited oral presentation at the 103rd meeting of the Acoustical Society of America in Chicago, Illinois on 27 April, 1982.

[Staal, 1983]

Staal, P.R. (1983). "Acoustic Propagation Measurements with a Bottom Mounted Array". In: *Acoustics and the Sea-bed* (N.G. Pace, ed.), pp. 289-296. Bath: Bath University Press.

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D		
(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)		
1. ORIGINATING ACTIVITY	2a. DOCUMENT SECURITY CLASSIFICATION	
Defence Research Establishment Atlantic	Unclassified	
	2b. GROUP	
	TM	
3. DOCUMENT TITLE		
Computer Control of the Underwater Acoustic Array Hydra		
4. DESCRIPTIVE NOTES (Type of report and inclusive dates)		
5. AUTHOR(S) (Last name, first name, middle initial)		
Staal, Philip, R.		
6. DOCUMENT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
September 1984	74	3
8a. PROJECT OR GRANT NO.	9a. ORIGINATOR'S DOCUMENT NUMBER(S)	
	DREA Technical Memorandum 84/S	
8b. CONTRACT NO.	9b. OTHER DOCUMENT NO.(S) (Any other numbers that may be assigned this document)	
10. DISTRIBUTION STATEMENT		
11. SUPPLEMENTARY NOTES	12. SPONSORING ACTIVITY	
13. ABSTRACT		
<p>↓</p> <p>This document describes the software used to control the underwater acoustic array Hydra. The Hydra array was developed at Defence Research Establishment Atlantic (DREA) for bottom mounted use in continental shelf waters. The array is controlled from a minicomputer on board ship, and through a microprocessor in the array. Both the minicomputer software and the microprocessor software are described.</p> <p>✓</p>		

1313
14-070

KEY WORDS

Computer Programs
Control Equipment
Hydrophone Arrays
Remote Control
Underwater Acoustics

INSTRUCTIONS

1. **ORIGINATING ACTIVITY:** Enter the name and address of the organization issuing the document.
- 2a. **DOCUMENT SECURITY CLASSIFICATION:** Enter the overall security classification of the document including special warning terms whenever applicable.
- 2b. **GROUP:** Enter security reclassification group number. The three groups are defined in Appendix M of the DRB Security Regulations.
3. **DOCUMENT TITLE:** Enter the complete document title in all capital letters. Titles in all cases should be unclassified. If a sufficiently descriptive title cannot be selected without classification, show title classification with the usual one-capital-letter abbreviation in parentheses immediately following the title.
4. **DESCRIPTIVE NOTES:** Enter the category of document, e.g. technical report, technical note or technical letter. If appropriate, enter the type of document, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.
5. **AUTHOR(S):** Enter the name(s) of author(s) as shown on or in the document. Enter last name, first name, middle initial. If military, show rank. The name of the principal author is an absolute minimum requirement.
6. **DOCUMENT DATE:** Enter the date (month, year) of Establishment approval for publication of the document.
- 7a. **TOTAL NUMBER OF PAGES:** The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.
- 7b. **NUMBER OF REFERENCES:** Enter the total number of references cited in the document.
- 8a. **PROJECT OR GRANT NUMBER:** If appropriate, enter the applicable research and development project or grant number under which the document was written.
- 8b. **CONTRACT NUMBER:** If appropriate, enter the applicable number under which the document was written.
- 9a. **ORIGINATOR'S DOCUMENT NUMBER(S):** Enter the official document number by which the document will be identified and controlled by the originating activity. This number must be unique to this document.
- 9b. **OTHER DOCUMENT NUMBER(S):** If the document has been assigned any other document numbers (either by the originator or by the sponsor), also enter this number(s).
10. **DISTRIBUTION STATEMENT:** Enter any limitations on further dissemination of the document, other than those imposed by security classification, using standard statements such as:
 - (1) "Qualified requesters may obtain copies of this document from their defence documentation center."
 - (2) "Announcement and dissemination of this document is not authorized without prior approval from originating activity."
11. **SUPPLEMENTARY NOTES:** Use for additional explanatory notes.
12. **SPONSORING ACTIVITY:** Enter the name of the departmental project office or laboratory sponsoring the research and development. Include address.
13. **ABSTRACT:** Enter an abstract giving a brief and factual summary of the document, even though it may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall end with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (TS), (SI), (C), (R), or (U).

The length of the abstract should be limited to 20 single-spaced standard typewritten lines; 7 1/4 inches long.
14. **KEY WORDS:** Key words are technically meaningful terms or short phrases that characterize a document and could be helpful in cataloging the document. Key words should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context.

END

FILMED

12-84

DTIC